Automotive Racing Video Data Logger

ECE445 Fall 2012

Design Review

TA: Igor Fedorov

October 1st, 2012

Contents

I. **Introduction**..... 3

II. **Design**..... 4
2.1 Block Diagrams..... 4
2.2 Block Descriptions..... 8
2.2.1 Overall Summary..... 8
2.2.2 Input Module..... 9
2.2.3 FPGA and Control Module..... 11
2.2.4 Storage Module..... 14
2.2.5 Display Module..... 18
2.2.6 Power Module..... 20
2.3 Schematics of Overall System..... 22
2.4 Simulations and Calculations.....

III. **Requirements and Verification**..... ?
3.1 Requirements & Verifications..... ?
3.1.1 Requirements Summary..... ?
3.1.2 Verifications Summary..... ?
3.1.3 Input Module..... ?
3.1.4 FPGA and Control Module.....
3.1.5 Storage Module.....
3.1.6 Display Module.....
3.1.7 Power Module.....
3.2 Tolerance analysis.....
3.3 Ethical Issues.....

IV. **Cost and Schedule**..... ?
4.1 Cost Analysis..... ?
4.2 Schedule..... ?
4.3 Contingency Plan.....

V. References..... ?

I. Introduction

1) Statement of Purpose:

The project goal is to develop a data logger video overlay that can be used in a car. The final product should be able to acquire near real-time engine data through the On-Board Diagnostic Port (OBDII) installed in a car. This product will also take video of a ride through the car's windshield with user supplied standard definition camera. Next, it will overlay OBD data onto the captured video and store it in a portable medium. The video stored in the media should be in a computer readable video format like JPEG2000. Raw OBDII data should also be stored as a file in the storage media. The driver can select playback video on the spot with overlay or replay the video with data at a time later.

This product will aid in automotive racing and tuning by being more intuitive for driver's to use and have better functions than existing products on the market.

2) Benefits and Features:

Currently, products like Dashware create a data video overlay AFTER a user uploads their video and data to a home computer. This feature is not suitable for amateur racers because they may need to review video and data immediately after a race. Also, this style of operation makes system set-up difficult because they cannot test their logging systems in real-time.

Our logger solves this problem with the abilities to:

- Record video and create an overlay in real-time and
- Use logger hardware to playback captured video.
- Capture standalone OBDII data

According to a few internet reviews, some users prefer to have a data logger record video with OBDII data overlayed on the screen, whereas other users prefer to have raw video recorded and race data recorded separately. Currently, there are products like Track vision that record the raw data onto an overlay but do not allow for separate video/OBDII streams.

Our product solves this problem by allowing the user to manually select a recording mode. They can pick either OBD II on video overlay raw video recorded with an associated OBDII data file for later analysis.

We will also implement an intuitive interface for drivers to use with mechanical switches. Interface with the data logger should be intuitive for fast set-up times and can be done by feel. Some products on the market require a computer to set-up the logger, which is not conducive for race conditions when time between race events is critical for success.

II. Design

2.1 Block Diagrams

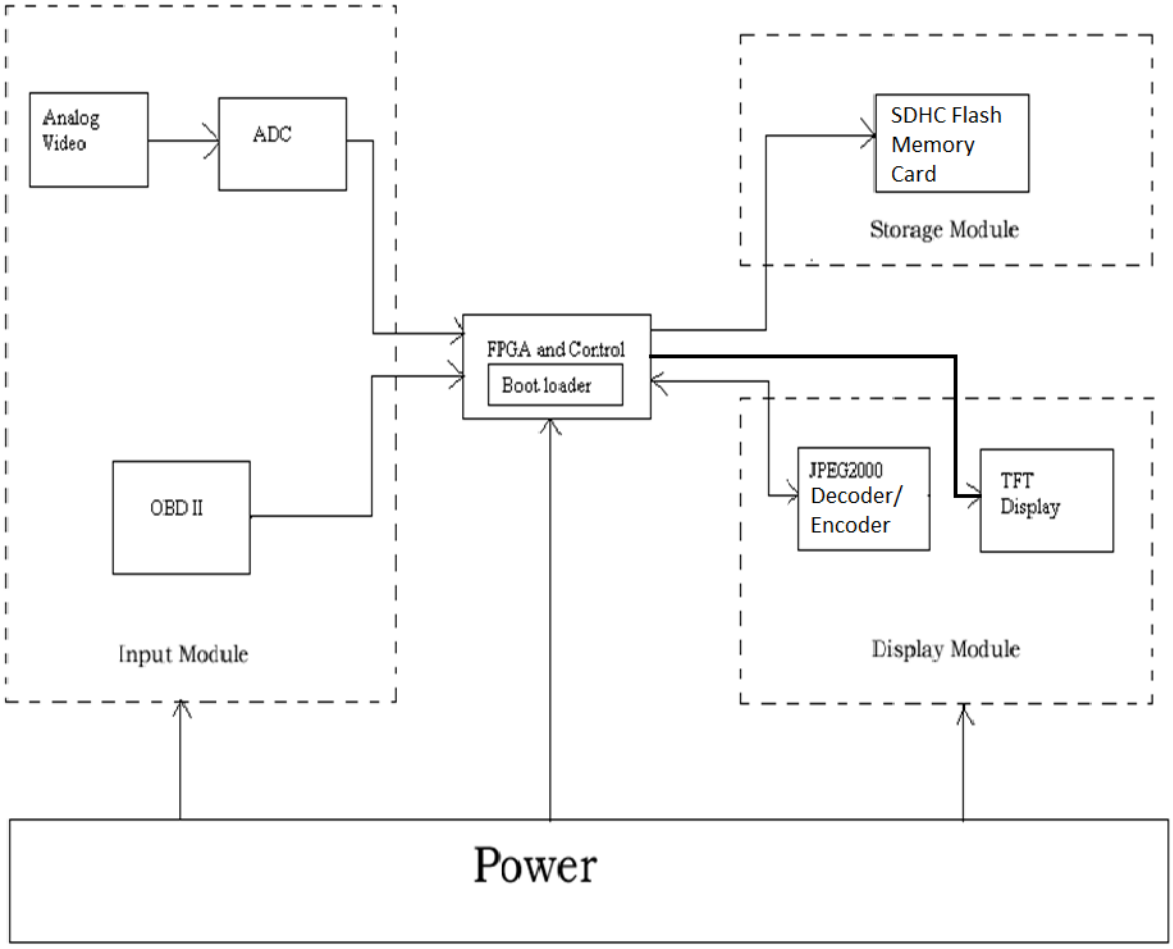


Figure 1: Top Level Block diagram

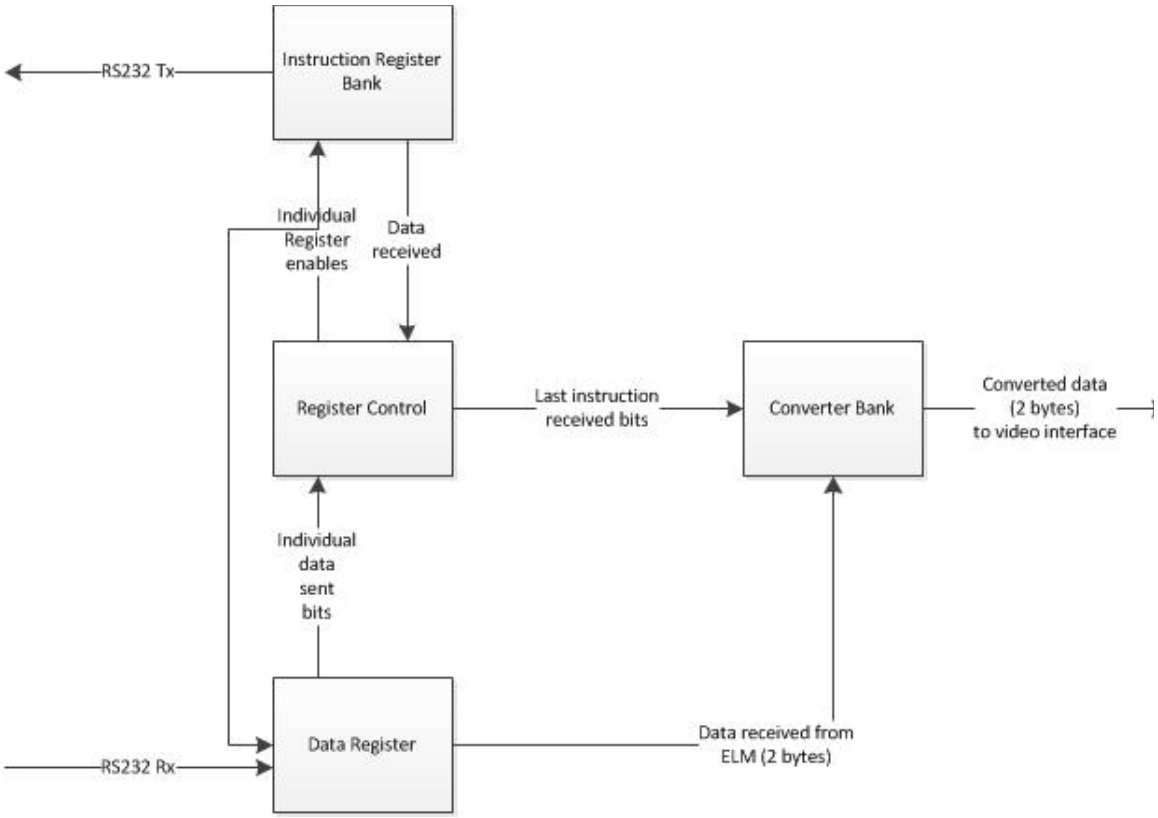


Figure 2: OBD II Block diagram

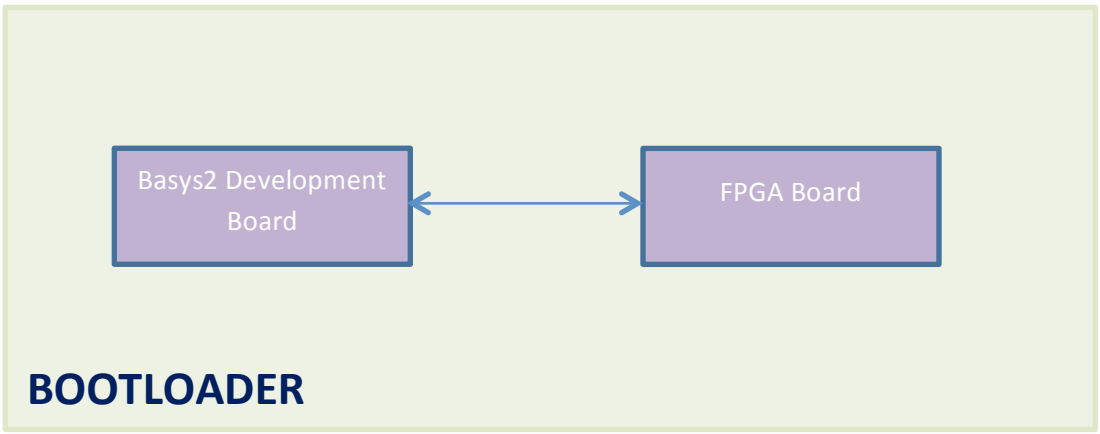


Figure 3: Bootloader Block diagram

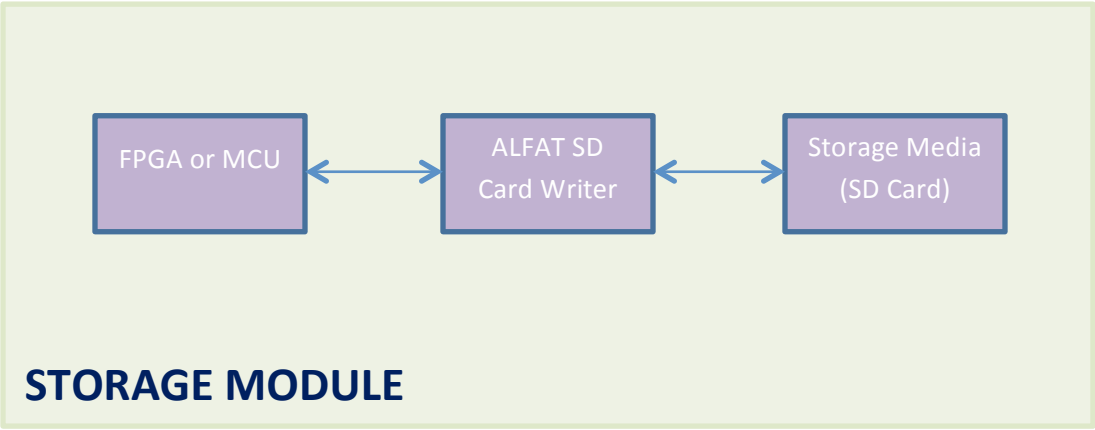


Figure 4: Storage Module Block diagram

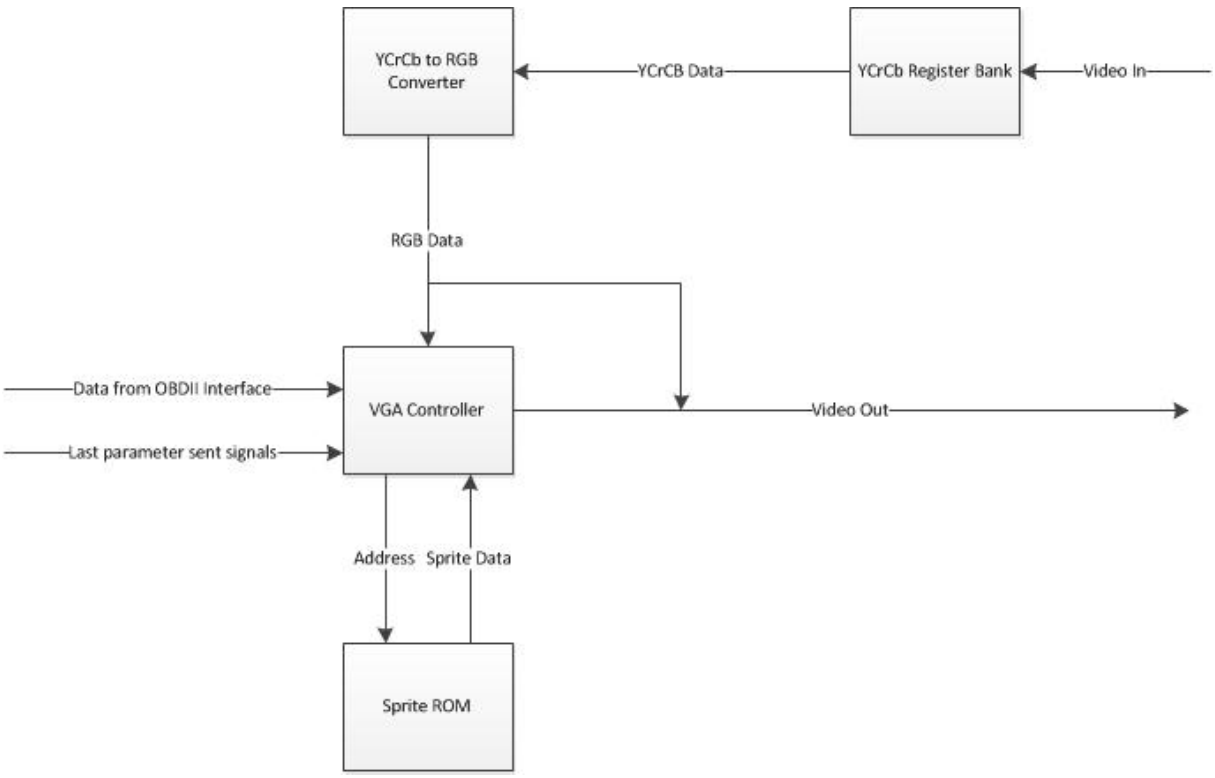


Figure 5: Video Display Module Block diagram

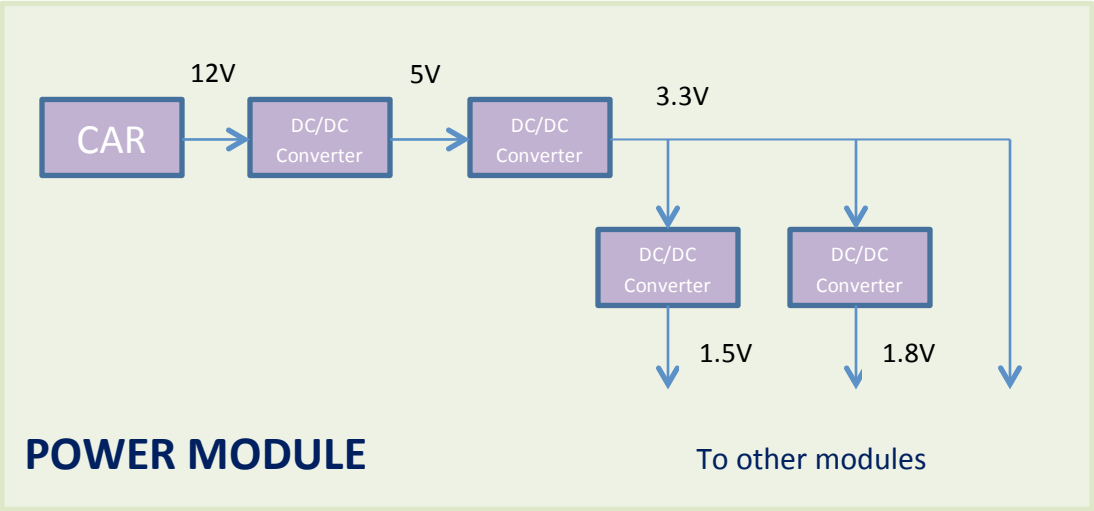


Figure 6: Power Module Block diagram

You have described each module and what it does, but you have not described the inputs and outputs of each model. In other words, you have not described how each module interfaces with the others. For example, you can write: "The power supply module receives an input from a user controlled switch and outputs a 5V DC supply voltage for the rest of the blocks. The power supply module contains a converter…."

2.2 Block Descriptions

2.2.1 Overall Summary:

1) **Input Data Module:** This module obtains input data such as information from the OBD II and the video images from the camera. The module then converts them into digital form using an ADC and transmits this digital data to the FPGA.
    a. **Analog Video:** The analog video obtained from the camera on the windshield. The image will be sent to the ADC to be converted into digital form.
    b. **ADC:** The Analog-to-Digital converter that converts standard definition analog video input into a digital form so it can be used by the FPGA. ADC input is analog video, which is converted to digital images and are then routed to the FPGA.
    c. **OBD II:** The On-Board Diagnostic system on a car is where information such as the MIL (malfunction indicator light), DTC (diagnostic trouble code), I/M (inspection and maintenance) info, etc. can be extracted from. We will focus on polling operating data like engine RPM, temperature, etc. The information obtained from this block is sent to the FPGA.
2) **FPGA and Control Module:** This module is the "brain" of the entire circuit. It receives the OBD II data and video images from the input module, processes them, overlays the data onto the video stream. It then routes the overlaid video to the storage and display modules. This entity also manages video stream compression/decompression. This block also boots the system.
    a. **Boot loader:** The boot loader is a nonvolatile memory circuit and co-processor that configures the FPGA for use when system is powered on.
    b. **JPEG2000 Decoder/Encoder:** A device that compresses/decompresses digital video for extended storage and computer access. This block receives processed video and data from the FPGA and routes decompressed output back to the FPGA so that video can be routed to other entities.
    c. **FPGA**: The integrated circuit that will be used to design the digital system to process the images and data obtained from the OBD II. This block obtains video from the camera and data from the OBD II, process them, and routes signal through a video compressor/decompressor. FPGA also streams processed to a storage device (the MMC) and controls data display for the attached TFT screen.
3) **Storage Module:** This module receives the overlaid video from the FPGA and control module and stores it on a storage device for later use. A compact flash MMC may be used as the storage device. User can also replay video from the storage medium.
    a. **SDHC Flash Memory Card**: The flash memory data storage device where the video and data obtained will be stored. This block obtains the processed video and data from the FPGA.
    b. **USB**: It is possible to achieve higher storage speeds with an FUSB PHY chip over USB interface. This item may be used depending on time remaining for project completion.
4) **Display Module:** This module displays the overlaid video on a LCD screen from storage through the FPGA/control block.
    a. **TFT Display:** The LCD display where the processed video obtained from the FPGA will be displayed.
5) **Power Module:** This module supplies power to the FPGA and Control Module.

2.2.2 Input Module

OBD II Interface:

Register Control
Description: This entity consists of a state machine that performs a loop that first initializes the ELM327 chip and then loops through a series of states that send and then received data from the ELM327. The states go from transmitting data, to receiving data, then back to sending, etc. First, an enable is sent to one of the instruction registers to allow a certain instruction to be sent to the OBDII. Then, once a signal is received from the register indicating that the data has been sent, the data register is enabled so as to receive the requested information from the OBDII. Once this data is received, a signal from the register is made high and sent to the register control, indicating that the data was received properly. Then the control moves to the next instruction and enables the appropriate register, but also sends a signal to the converter bank and the video interface indicating what the last received information was, and enabling the correct conversion entity depending on that data. Note that only one register enable and one converter enable are ever high at any given time to ensure that only one instruction is sent or received at a time and that only one parameter is being converted at a time to ensure that no data gets garbled and to ensure that all the data is sent and received in a clear order.

Instruction Register Bank
Description: When a register is first initialized, a 4 or 6 byte data signal is set, depending on the instruction being sent. If the enable is active, then the most significant bit of the data signal is outputted. Then, the signal is shifted to the left by one bit, and a counter increases by one. This continues until the counter reaches the number of bits in the instruction. Then, the signal going to the control register becomes active, indicating that the message has been sent. This causes the state in the control register to change, causing the enable to go low and the register to reinitialize to its original value.

Instructions we plan to use:
x0105: Engine coolant temperature
x010A: Fuel pressure
x010C: Engine RPM
x010D: Vehicle speed
x010F: Intake air temperature
x0111: Throttle position
x0146: Ambient air temperature
x015C: Engine oil temperature

Note: These may be changed depending on parameters available for the car we test. The one we use may not be compatible with some of the instructions listed.

Data Register
Description: When enabled, this register receives data from the OBDII one bit at a time. When a bit is received, the register shifts the bits to “make room” for the next one. Also, a counter is used to keep track of the amount of data received. Once all of the data is received, a signal is sent to the register control to indicate such. Notice that the counter counts to different values depending on whether one or two bytes are being received.

Converter Bank

Description: When enabled, the converter uses an arithmetic equation to change the data received into the actual, factual information from the engine. The equations used are explained in the calculations section. These new values are then sent to the video interface. Note that if the converter is not enabled, its output is set to high impedance, so that only one output is being transmitted at any time.

2.2.3 FPGA and Control Module

a. Boot loader

According to the Cyclone II Device Handbook, Cyclone II devices use SRAM cells to store configuration data. Because SRAM memory is volatile, all configuration data will be lost once the device powers off. Therefore, there's a need to implement a boot loader to download configuration data to the Cyclone II devices every time the device powers up.
In our design, we'll use the Digilent Basys2 development board as the FPGA's boot loader.
The Cyclone II (or the EP2C35 processor) supports 3 configuration schemes: AS (active serial), PS (passive serial) and JTAG-based configuration. To select a configuration scheme, the MSEL [ ] must be connected to the appropriate digital logic.

Table 13–1. Cyclone II Configuration Schemes

Configuration Scheme	MSEL1	MSEL0
AS (20 MHz)	0	0
PS	0	1
Fast AS (40 MHz) *(1)*	1	0
JTAG-based Configuration *(2)*	*(3)*	*(3)*

Figure 7: Cyclone II Configuration Schemes (Cyclone II Device Handbook, Volume 1, p. 356)

For our design, the AS scheme will be used, since it has the ability to configure the device at a high frequency (40MHz).

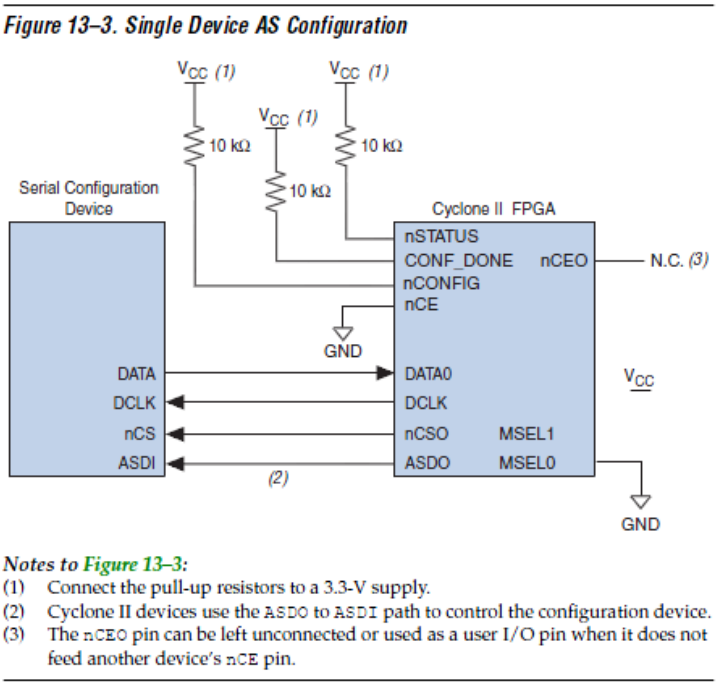


Figure 8: Single Device AS Configuration (Cyclone II Device Handbook, Volume 1, p. 362)

The EP2C35 essentially goes through 3 states (RESET, CONFIGURATION and INITIALIZATION) before USER MODE, as depicted below,

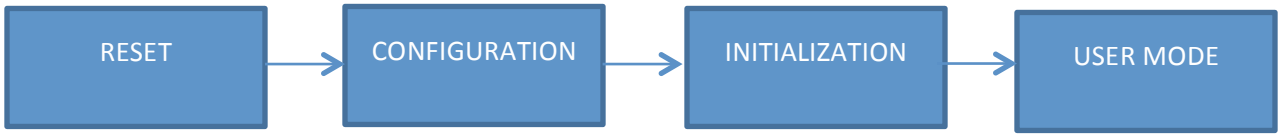
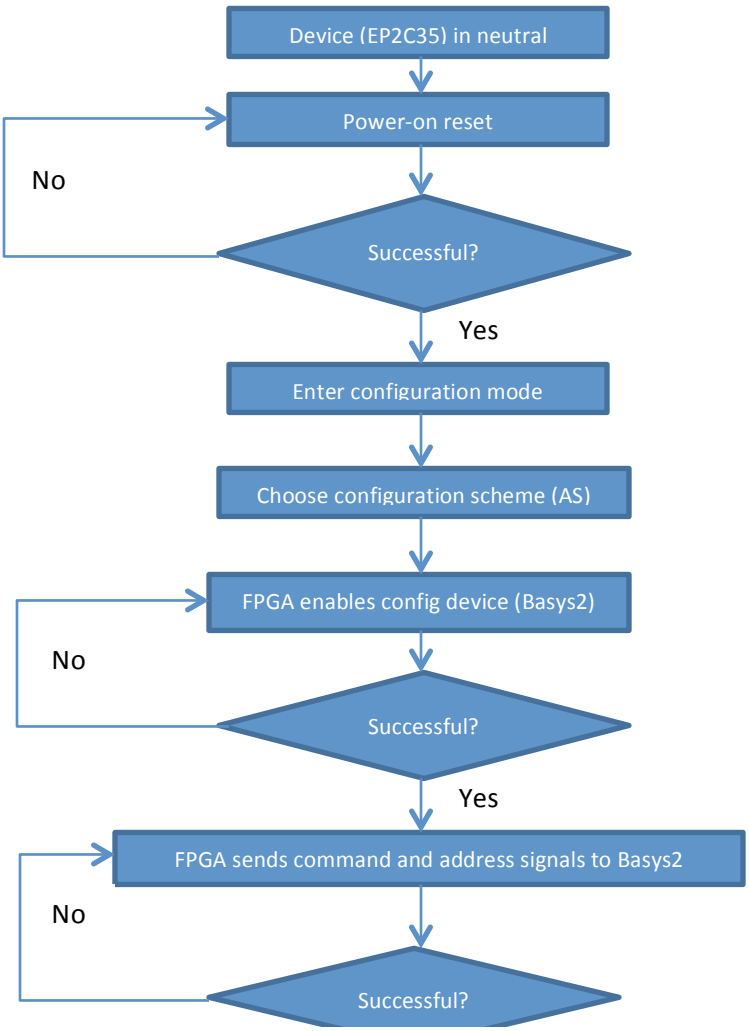


Figure 9: Cyclone II device configuration states

The EP2C35configuration flowchart is as follows.



2.2.4 Storage Module

After the video is processed with the overlaid data, it will be stored on a storage device. We'll choose the Transcend 16 GB SDHC Class 10 Flash Memory Card as our storage device. To simplify the process of writing the file to disk (which involves writing complex memory card and USB drivers and manipulate buggy file system), we'll use the ALFAT OEM Board – FAT32 SD card writer from GHI Electronics.

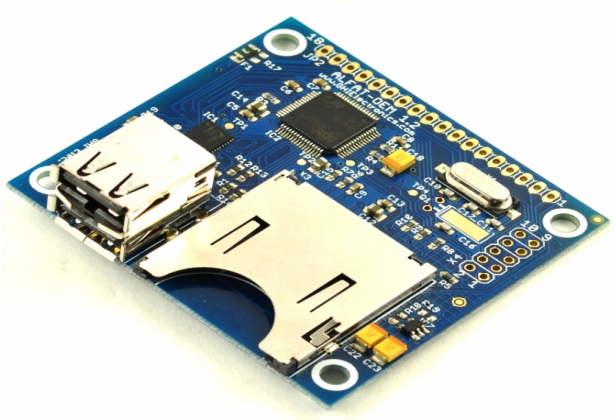


Figure 10: ALFAT OEM Board (ALFAT SoC Processor User Manual, p. 45)

The ALFAT SoC processor gives the FPGA a simple way to access storage medias such as SD cards and USB Mass Storage devices in a very short time. Its key features are as follows:

- LFN (Long File Name) supports.
- FAT16 and FAT32 systems.

- Friendly user-interface through UART, SPI, I2C.
- No limits on media size, file size, or file/folder count.
- Up to 8 simultaneous file access.

For our design, we choose to implement it using the UART interface. A simple schematic of how the ALFAT SoC interacts with other devices is as follows,

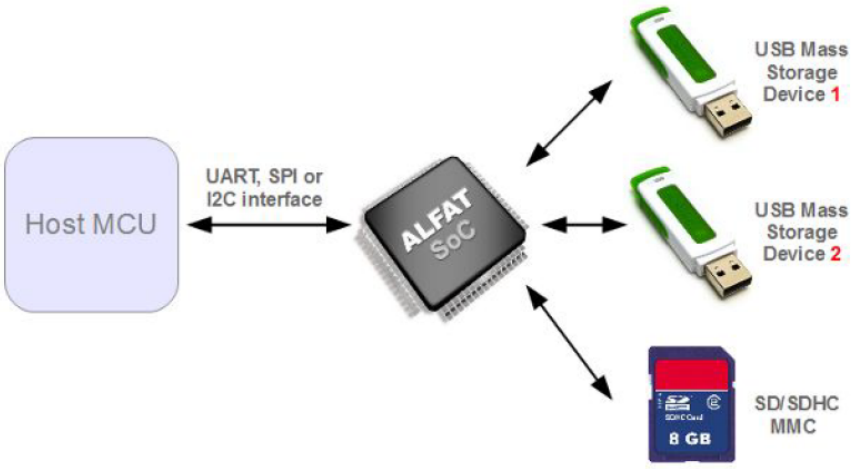


Figure 11: ALFAT SoC interface (ALFAT SoC Processor User Manual, p. 5)

After power-up, the Host MCU (our FPGA board) controls the ALFAT SoC by sending in commands in human-readable ASCII format. The command list is as follows,

Command	Description	Command	Description
V	Get version number	I	Initialize MMC/SD or USB
Z	Low power	O	Open file to a free handle
T	Initialize Real Time Clock	W	Write to a file
S	Set current time and date	R	Read from a file
G	Get current time and date	F	Flush file
B	Change Baud rate	C	Close file
#	Enable echo	P	File seek
J	Read status register	Y	File tell
E	Test media	D	Delete file or folder
K	Get free size	?	Find file or folder
@	Initialize directory list	M	Copy From File to Another
N	Get next directory entry	A	Rename file
Q	Format	L	Fast Write to file (SPI only)

Figure 12: ALFAT Command Set (ALFAT SoC Processor User Manual, p. 19)

The command will be send via a UART interface, which uses 3 hardware signals,

- UART_TX signal to send data out from ALFAT
- UART_RX signal to receive data to ALFAT
- UART_BUSY signal which should be monitored while sending data to ALFAT. When hi, no more data should be transmitted to ALFAT until it gets lo.

Besides, commands are terminated with line-feed, and the user must read back the responses for each command properly and check whether the command was successful.
To write a file to a storage device, we need the following commands:

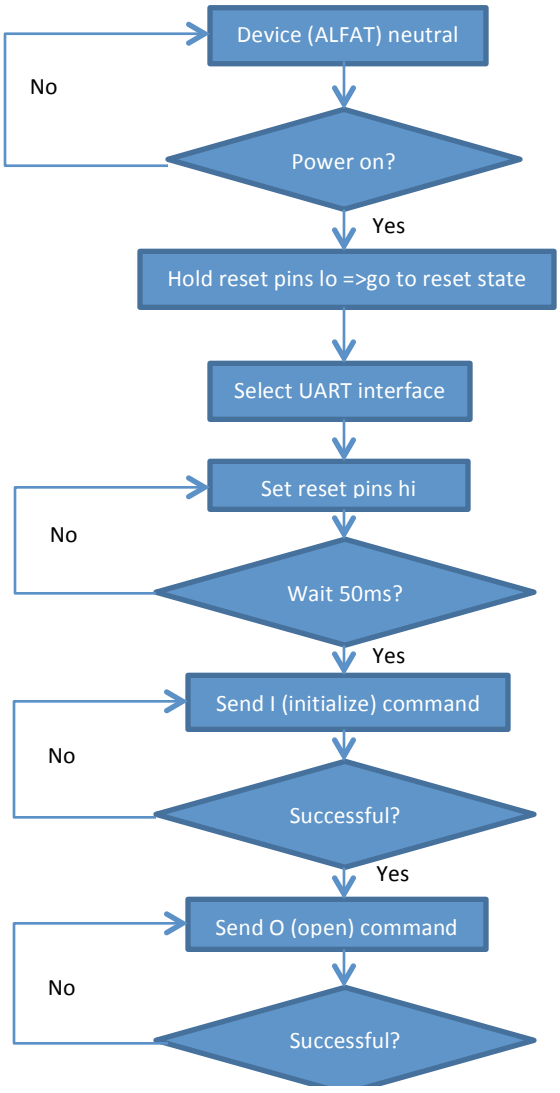
- I – Initialize and mount the storage device. If this command is not called first, the file system can be corrupted, and other file related commands will fail.
- O – Open file for Read, Write and Append, and give write privileges to it. If the file already existed, it will be erased and rewritten.
- W – Write to file through a file handle assigned to an open file with write mode. This command is accomplished through 3 steps:
  1) Send W command with file handle and the data size
  2) Wait til you get the acknowledge
  3) Send the data

  To make sure the data is written to a file, the file must be flushed or closed when done or there will be a risk of losing data or corrupting the file system if the storage media was removed or if there was a power loss.

- F – Flush the data of an opened file. This command is useful to make sure all the data are physically saved in the media.
- C – Close file. This command issues a flush internally and then release the file handle.

The flowchart for a WRITE operation is as follows,

To store a file on an SD card:



2.2.5 Display Module (Software)

Video is inputted into the module from the ADC chip as luma and chroma data, which is stored in a register. This data is then converted to the RGB color space and is sent to the VGA Controller. The data from the OBDII module is also sent to the VGA Controller, which uses a ROM to call upon and display the correct character sprites. Both the raw RGB video data and that of the video with overlaid data are outputted.

YCrCb Register Bank

Description: 1 byte is sent from the ADC at one time, but the information changes between the luma value and the chroma value, which is two bytes. There are 3 registers then to hold the last value sent for each of those. There is also a counter that controls the enable for each register, which is timed by the input clock. Each of these registers then outputs that byte to the YCrCb to RGB converter.

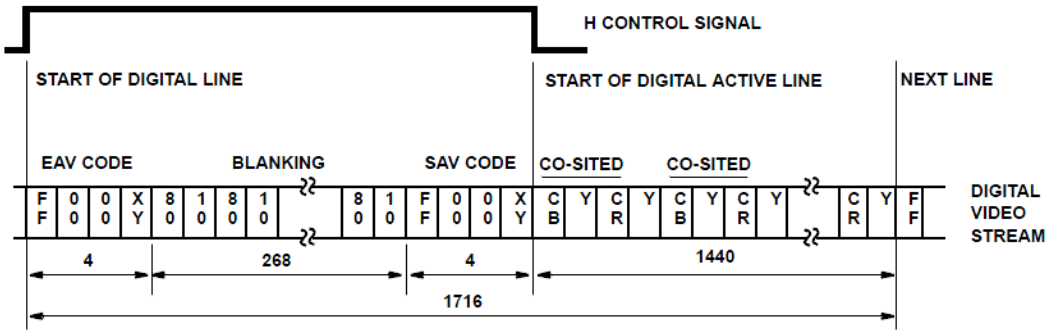


FIGURE 1. BT.656 8-BIT PARALLEL INTERFACE DATA FORMAT FOR 525/60 VIDEO SYSTEMS

Figure 13: BT.656 8-bit parallel interface data format for 525/60 video systems (from AN9728 Application Notes)

YCrCb to RGB Converter

Description: the Y, Cr, and Cb values are all constantly inputted. Then, they are converted into the RGB color space. The conversion calculations are explained in the next section. Once the conversions have been performed, the RGB values are sent to the VGA Controller as 3 separate bytes.

VGA Controller

Description: The VGA Controller will output information for each individual pixel going from left to right, then up to down. It will output the input for most of these pixels. However, at some points this entity will call upon the Sprite ROM, which then sends back 8 data bytes, each corresponding to 8 pixels on a single row of pixels. For each pixel, if the bit from the Sprite ROM is low, then the video outputted will simply be the RGB value inputted. However, if the value from the ROM is high, then the pixel will be changed to a different color. This will create characters on the screen. Some of these will be constant to describe the parameters, such as RPM, air temperature, etc. Others will be determined by the integer values of the data sent from the OBDII. These will be the numeric characters, and an if/then statement will be used to display the appropriate character.

Sprite ROM

Description: This entity acts as a simple read-only ROM, which outputs prewritten data to the VGA, depending on the address requested. One address is read from at a time, and one address is outputted at one time. Each address and each line of data are both 1 byte each.

2.2.6 Power Module

The Automotive Racing Video Data Logger device gets power from the 12V car battery. However, 12V is higher than needed for our device, so DC/DC voltage converters will be included in the power module to step down 12V to other useful voltages.

Specifically, here are the power requirements for all the components from the device:

Devices	Voltage(s) needed
Digilent Basys2 Development Board	3.5V-5.5V
ALFAT OEM Board	3.3V
FPGA board	3.3V, 1.8V, 1.5V, 1.2V
TFT LCD display	5V

From the 12V car battery, the voltage will be stepped down to 5V, and then 5V will be stepped down one more time to get 3.3V to supply power to the ALFAT OEM Board, the FPGA Board, etc.

3.3V will then be stepped down again to 1.8V and 1.5V to supply power to logic signals on the FPGA board. The power module block diagram from section 2.1 is copied below to illustrate how the car power supply will be converted to use on the device:

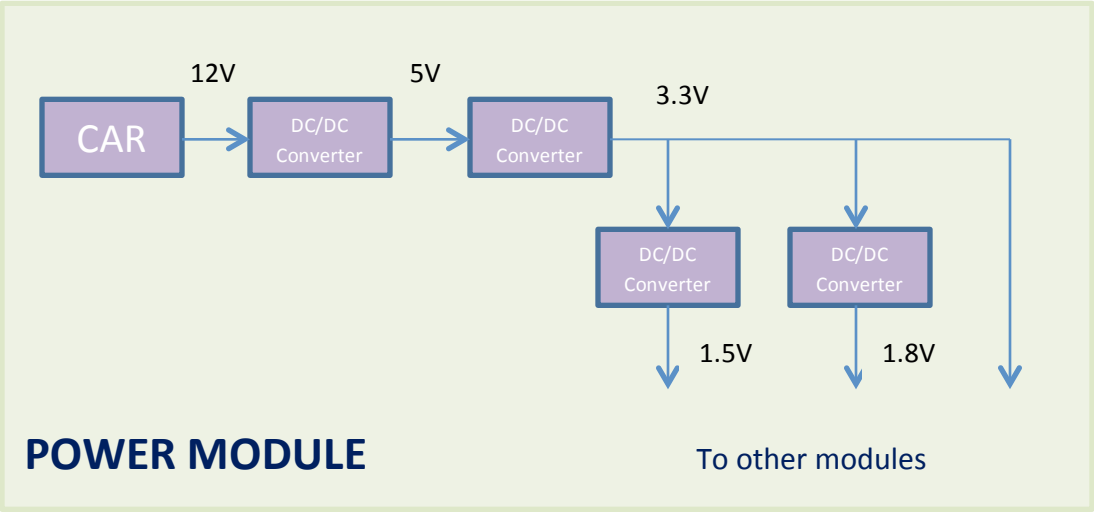
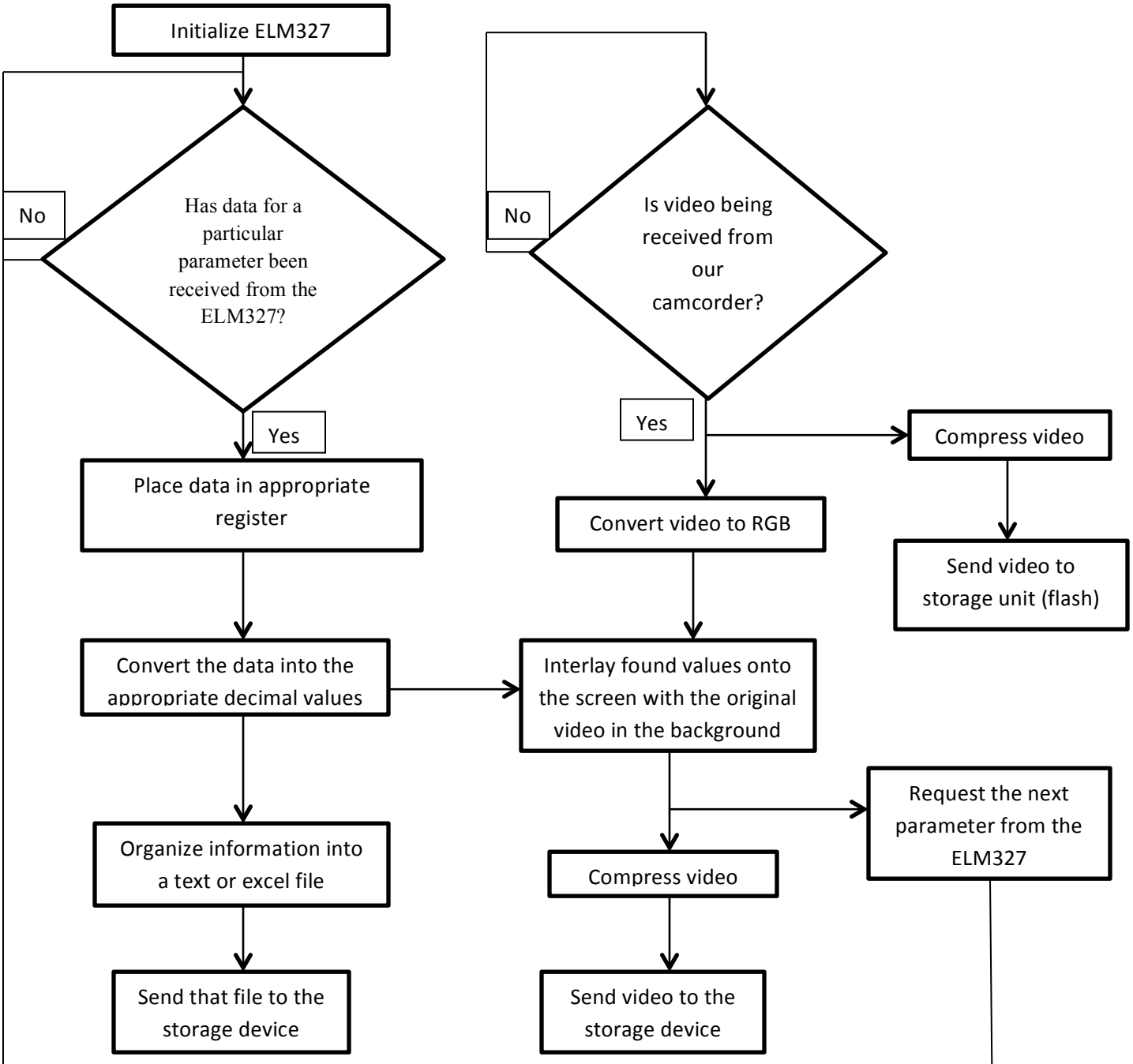


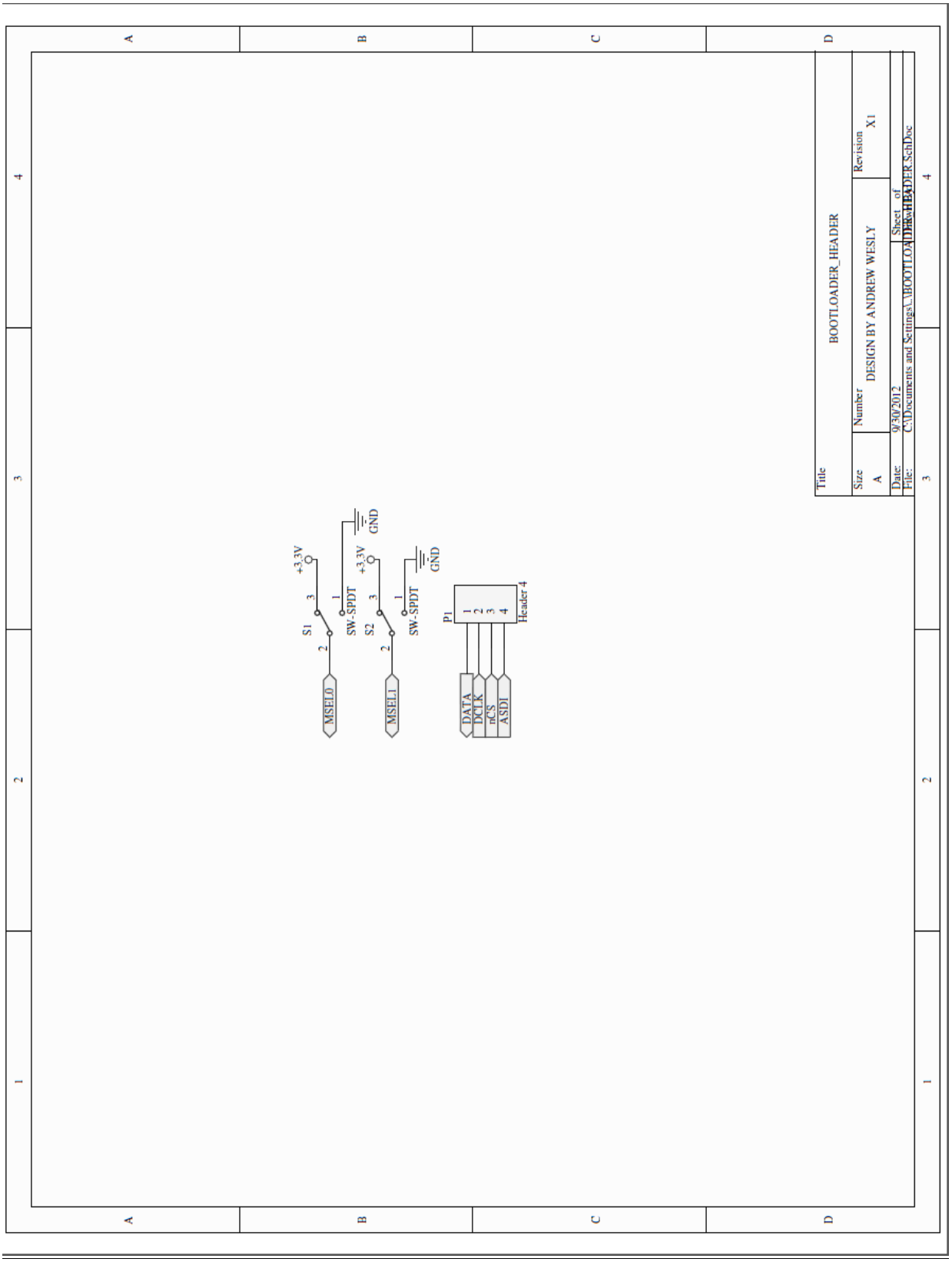
Figure 14: Power Module Block diagram

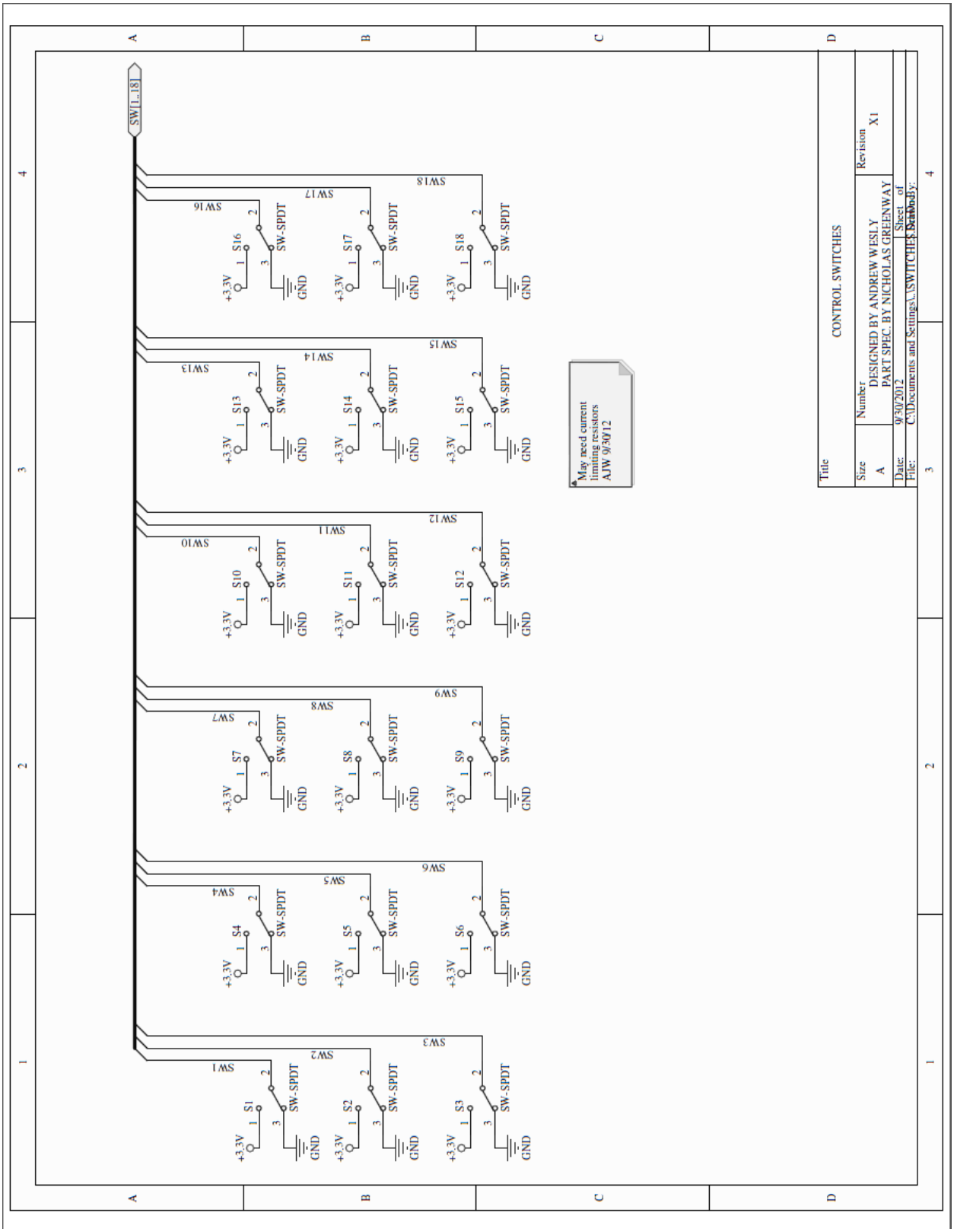
Overall the digital flowchart of the system is as follows,

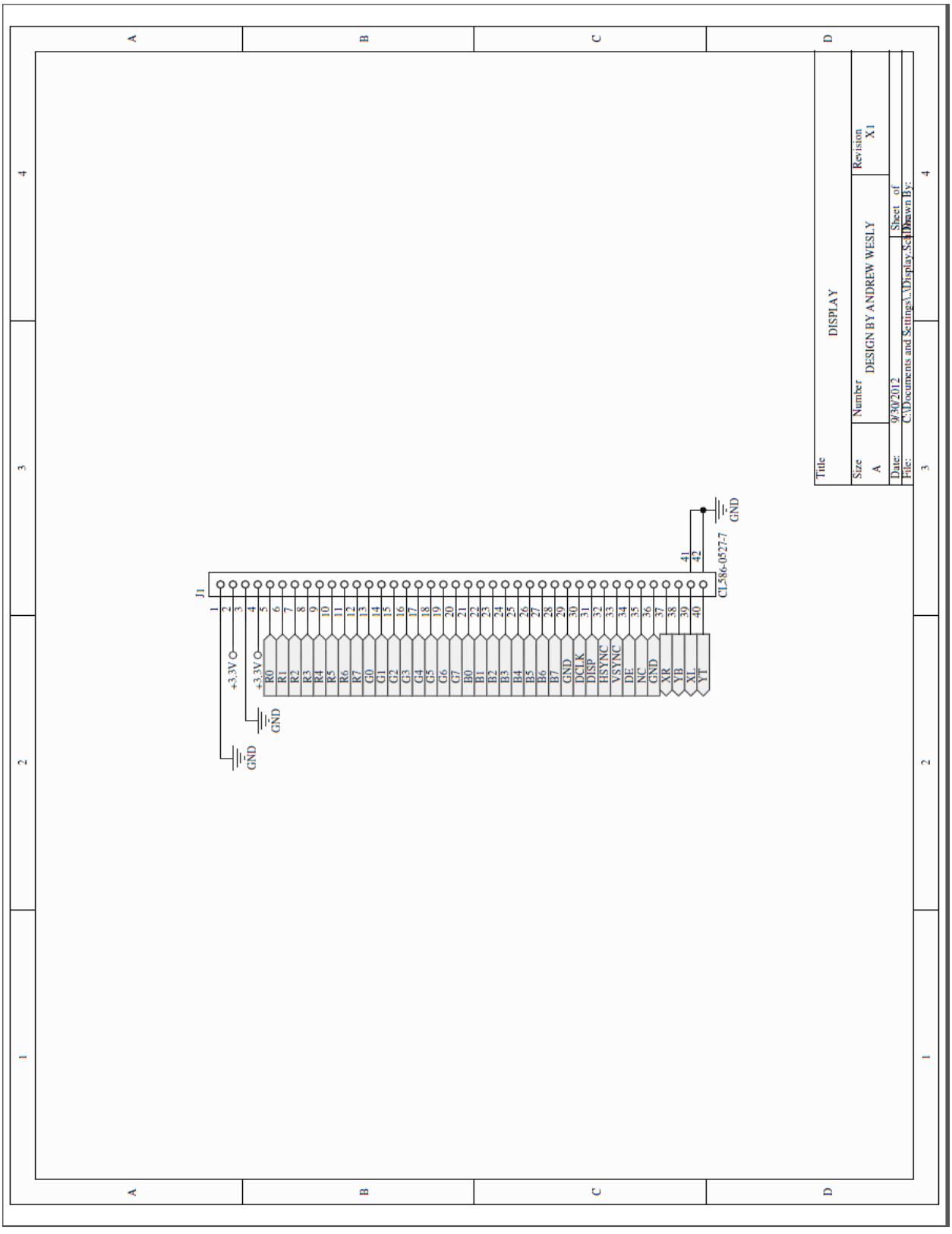

Initialize ELM327
Has data for a particular parameter been received from the ELM327?
No
Yes
Place data in appropriate register
Convert the data into the appropriate decimal values
Organize information into a text or excel file
Send that file to the storage device
Is video being received from our camcorder?
No
Yes
Compress video
Send video to storage unit (flash)
Convert video to RGB
Interlay found values onto the screen with the original video in the background
Request the next parameter from the ELM327
Compress video
Send video to the storage device

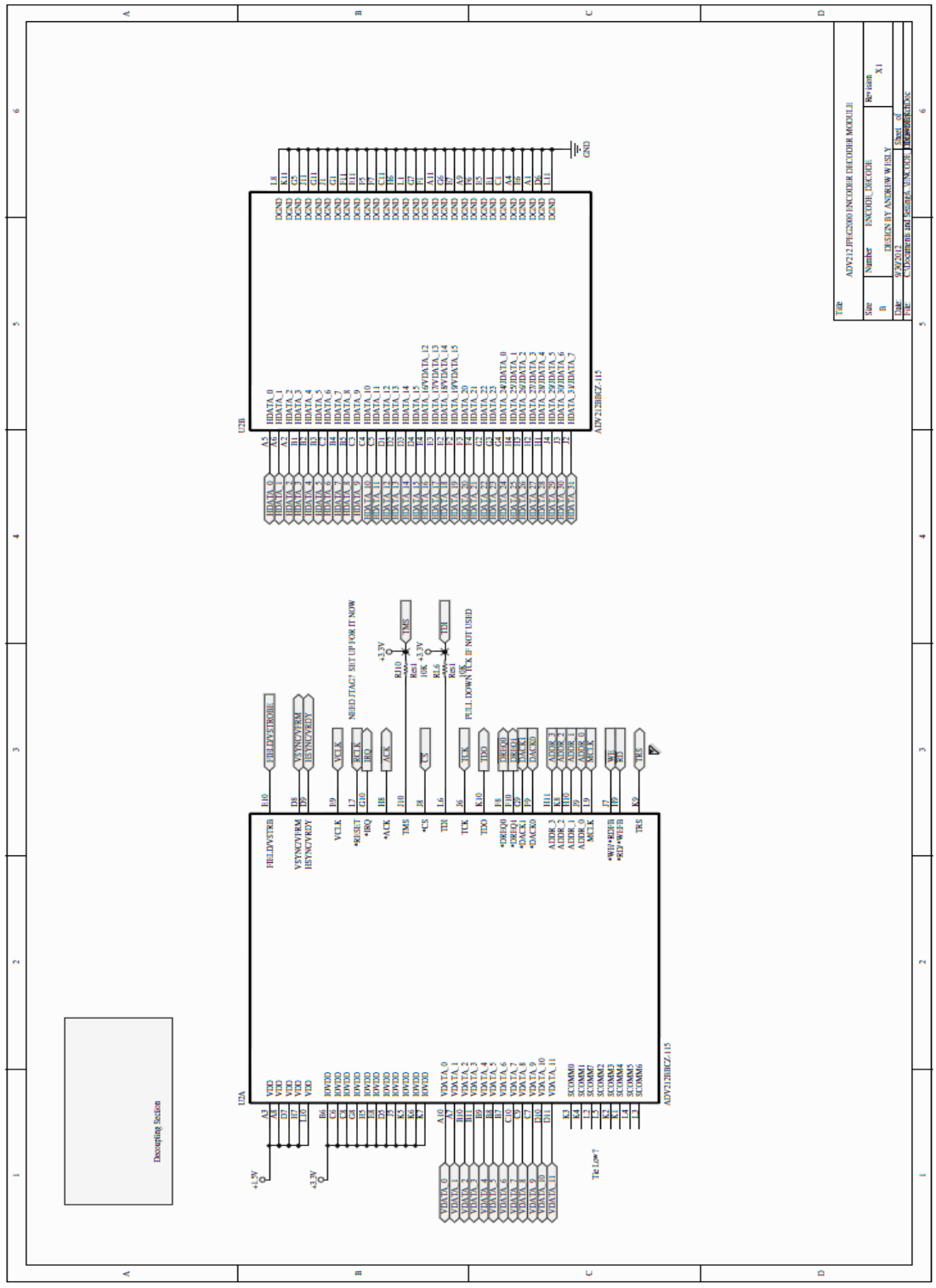
2.3 Figure 15: Overall Digital Flowchart

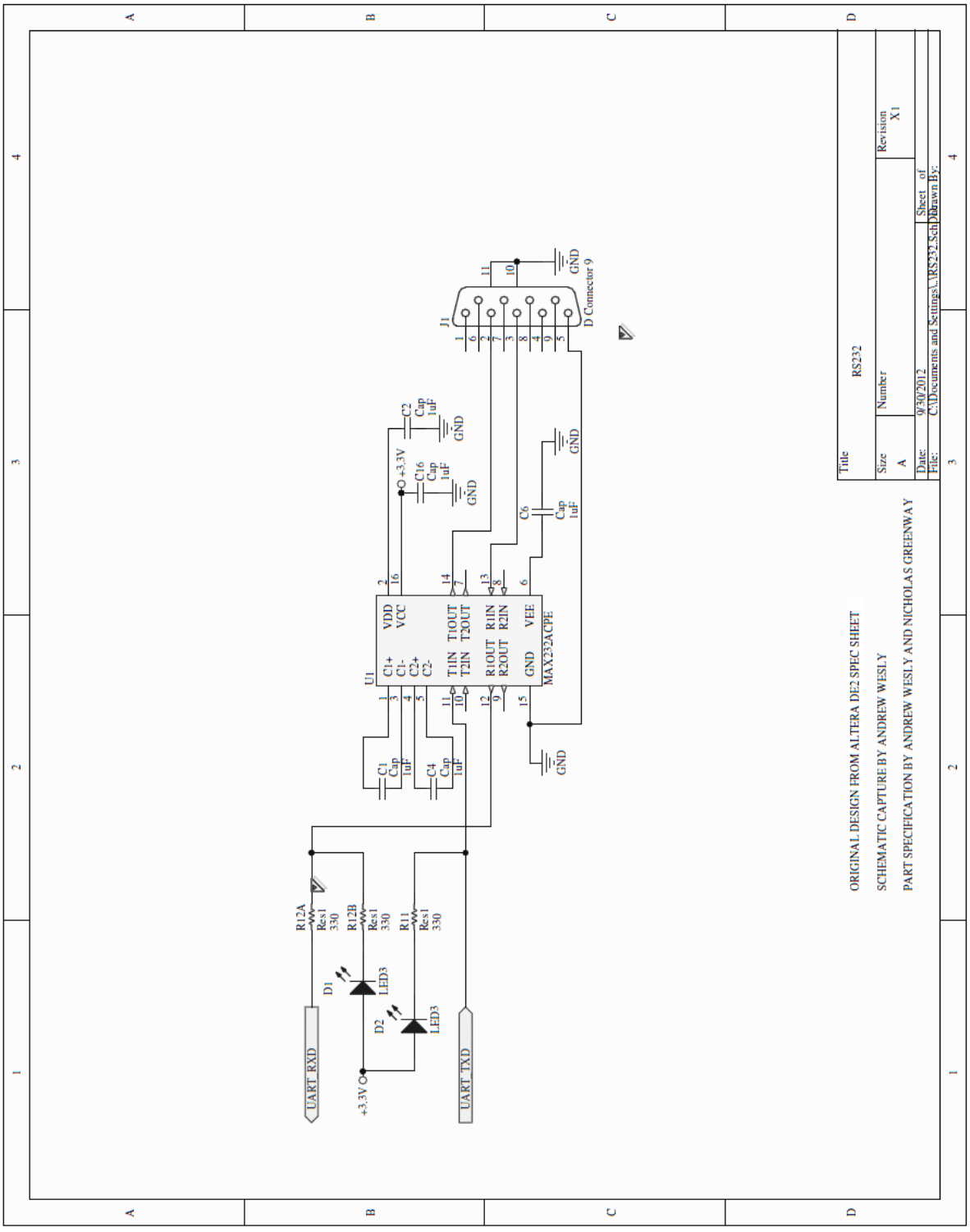
Your figures and diagrams need to be numbered and labeled with a description. As they are now, it is unclear to the reader exactly what is inside each figure.

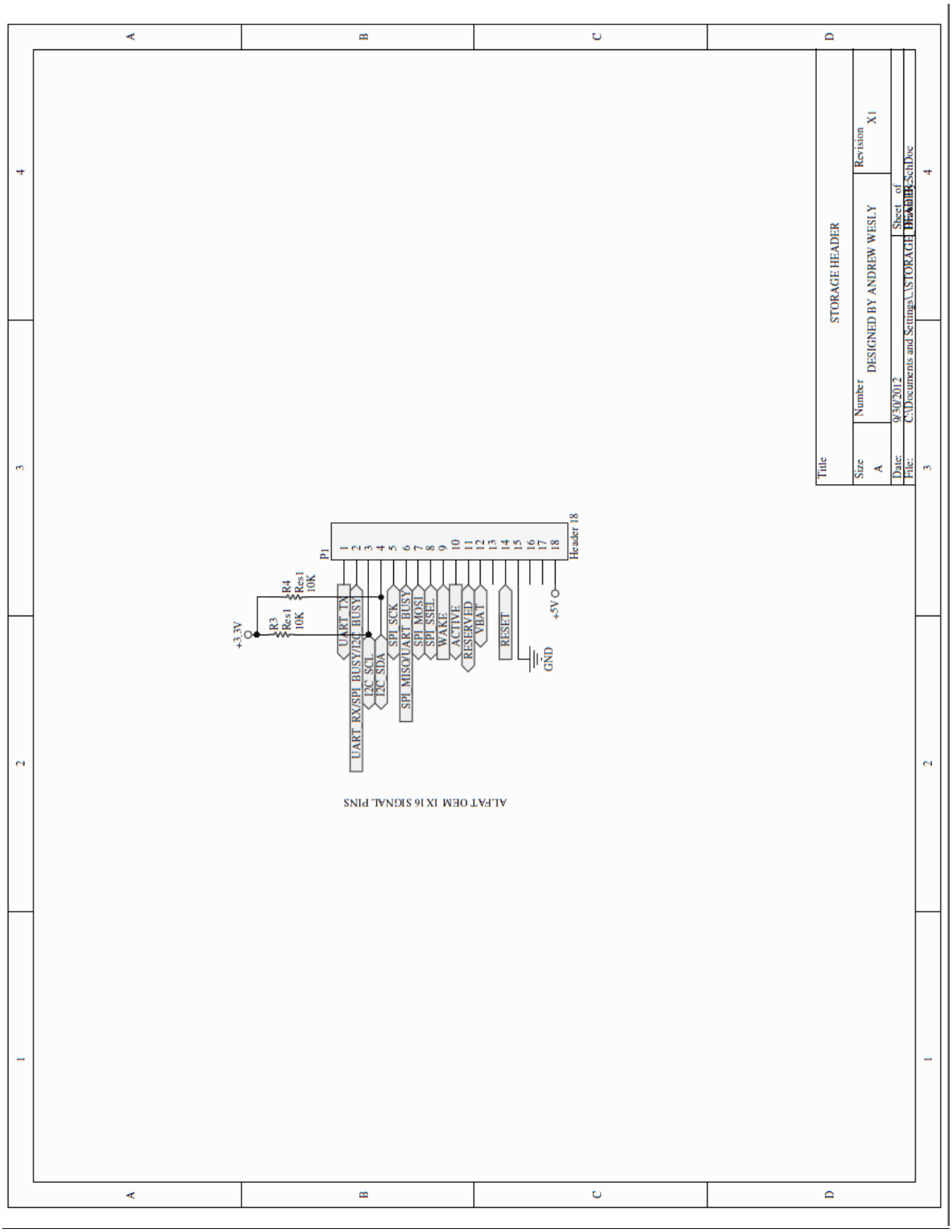


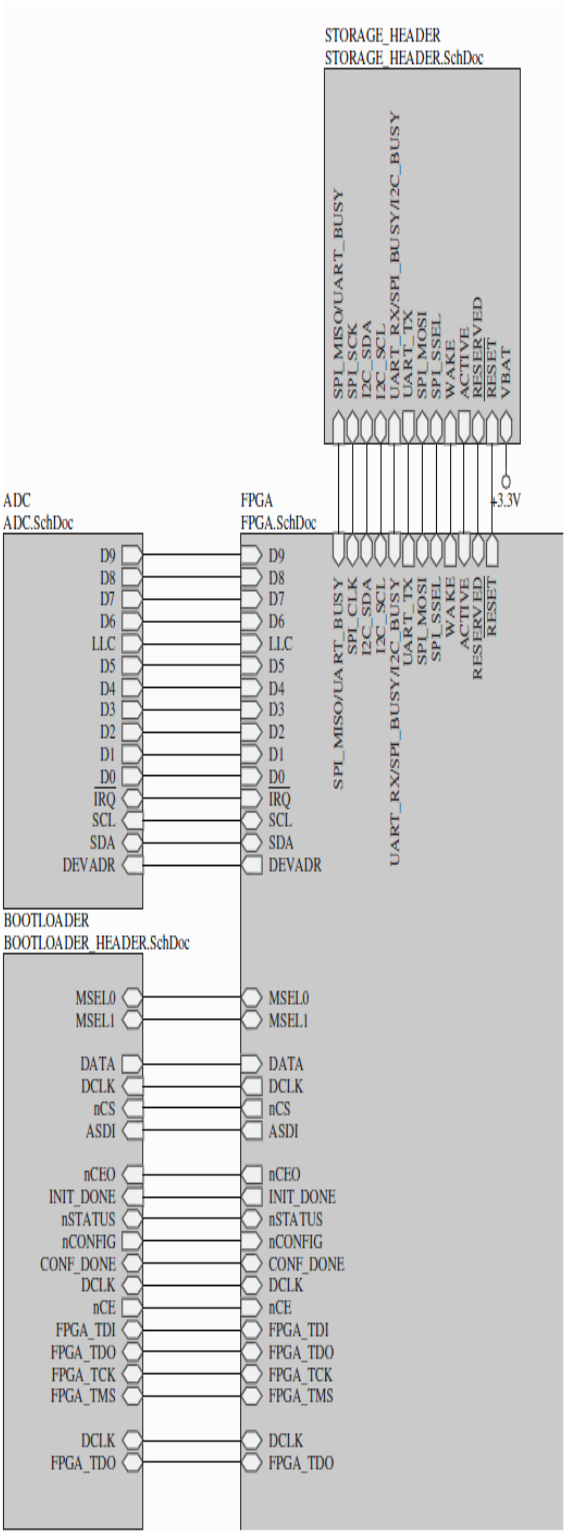

SW[1..18]
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
SW16
SW17
SW18
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
SW-SPDT
+3.3V
GND
May need current limiting resistors AJW 9/30/12
Title
CONTROL SWITCHES
Size
A
Number
DESIGNED BY ANDREW WESLY
PART SPEC. BY NICHOLAS GREENWAY
Revision
X1
Date:
9/30/2012
Sheet of
File:
C:\Documents and Settings\...\SWITCHES.SchDoc
Drawn By:

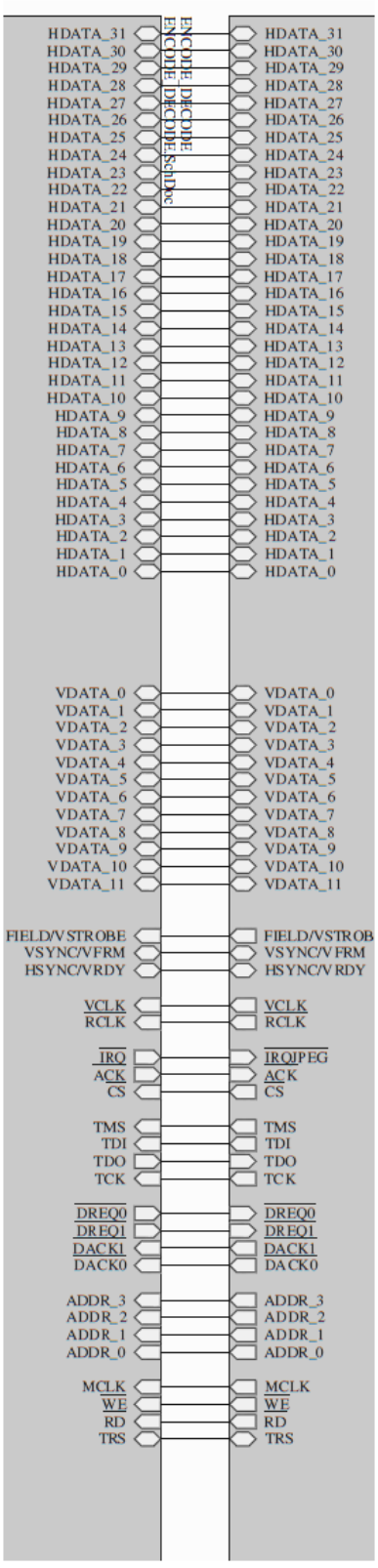

DISPLAY
Title
Size
A
Number
DESIGN BY ANDREW WESLY
Revision
X1
Date:
9/30/2012
Sheet of
File:
C:\Documents and Settings\...\Display.SchDoc
Drawn By:
J1
CL586-0527-7
+3.3V
+3.3V
GND
GND
GND
R0
R1
R2
R3
R4
R5
R6
R7
G0
G1
G2
G3
G4
G5
G6
G7
B0
B1
B2
B3
B4
B5
B6
B7
GND
DCLK
DISP
HSYNC
VSYNC
DE
NC
GND
XR
YB
XL
YT

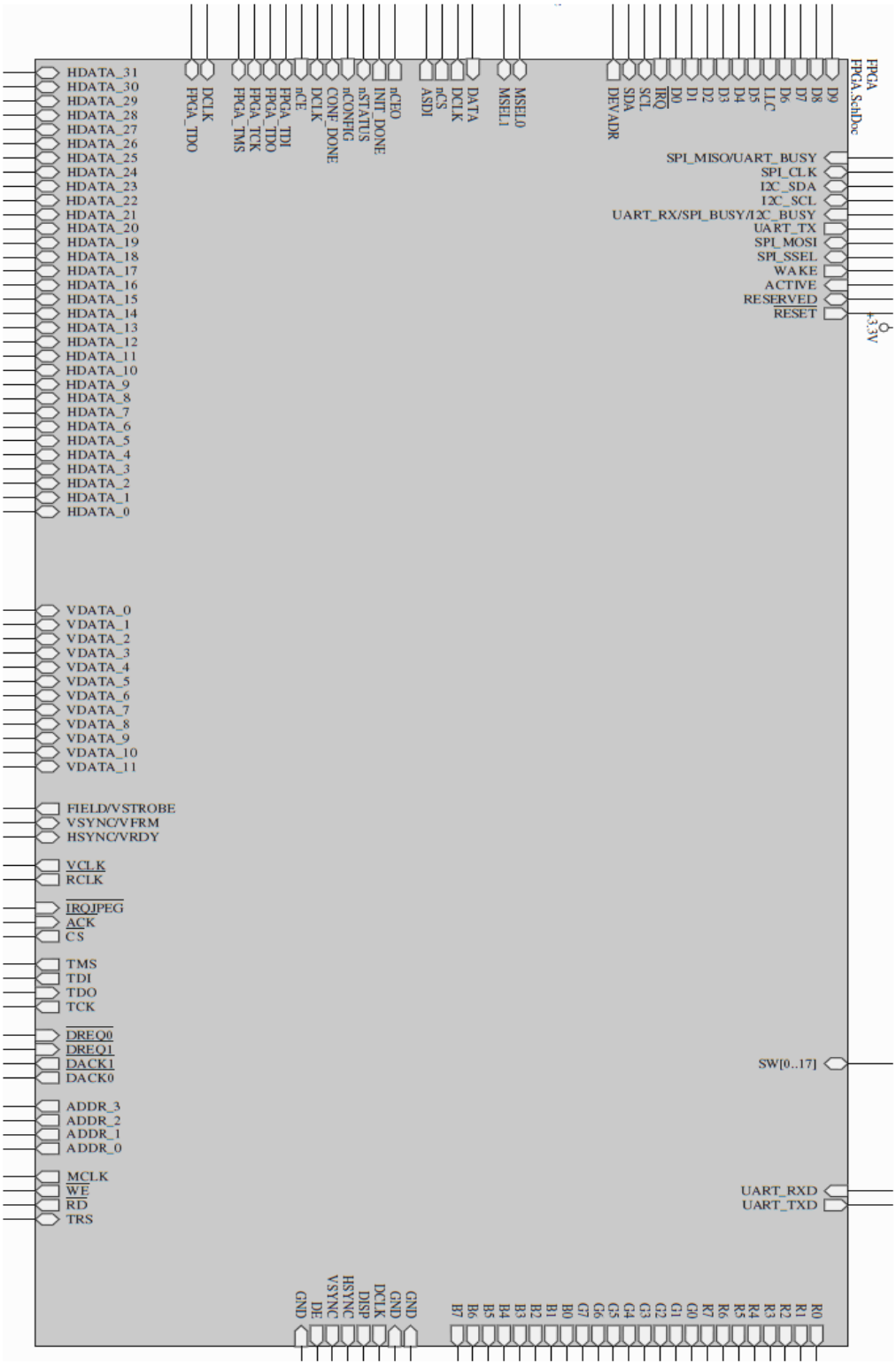

Decoupling Section
U2A
ADV212BBCZ-115
U2B
ADV212BBCZ-115
Title
ADV212 JPEG2000 ENCODER DECODER MODULE
Size
B
Number
ENCODE_DECODE
Revision
X1
DESIGN BY ANDREW WISE
Date:
9/9/2013
File:
C:\ANDREW\ 2012 Software\ENCODE_DECODE\ENCODE_DECODE.SchDoc
Sheet of
NEED JTAG? SET UP FOR IT NOW
PULL DOWN VCLK IF NOT USED
Tie Low?
GND


ORIGINAL DESIGN FROM ALTERA DE2 SPEC SHEET
SCHEMATIC CAPTURE BY ANDREW WESLY
PART SPECIFICATION BY ANDREW WESLY AND NICHOLAS GREENWAY
Title RS232
Size A
Number
Revision X1
Date: 9/30/2012
File: C:\Documents and Settings\...\RS232.SchDoc
Sheet of
Drawn By:
U1
MAX232ACPE
VDD VCC C1+ C1- C2+ C2- T1IN T2IN R1OUT R2OUT GND T1OUT T2OUT R1IN R2IN VEE
J1
D Connector 9
C1 Cap 1uF
C2 Cap 1uF
C4 Cap 1uF
C6 Cap 1uF
C16 Cap 1uF
+3.3V
GND
R12A Res1 330
R12B Res1 330
R11 Res1 330
D1 LED3
D2 LED3
UART RXD
UART TXD


STORAGE HEADER
DESIGNED BY ANDREW WESLY
Header 18
P1
+3.3V
R3
Res1
10K
R4
Res1
10K
UART_TX
UART_RX/SPI_BUSY/I2C_BUSY
I2C_SCL
I2C_SDA
SPI_SCK
SPI_MISO/UART_BUSY
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
VBAT
RESET
GND
+5V
ALFAT OEM 1X16 SIGNAL PINS

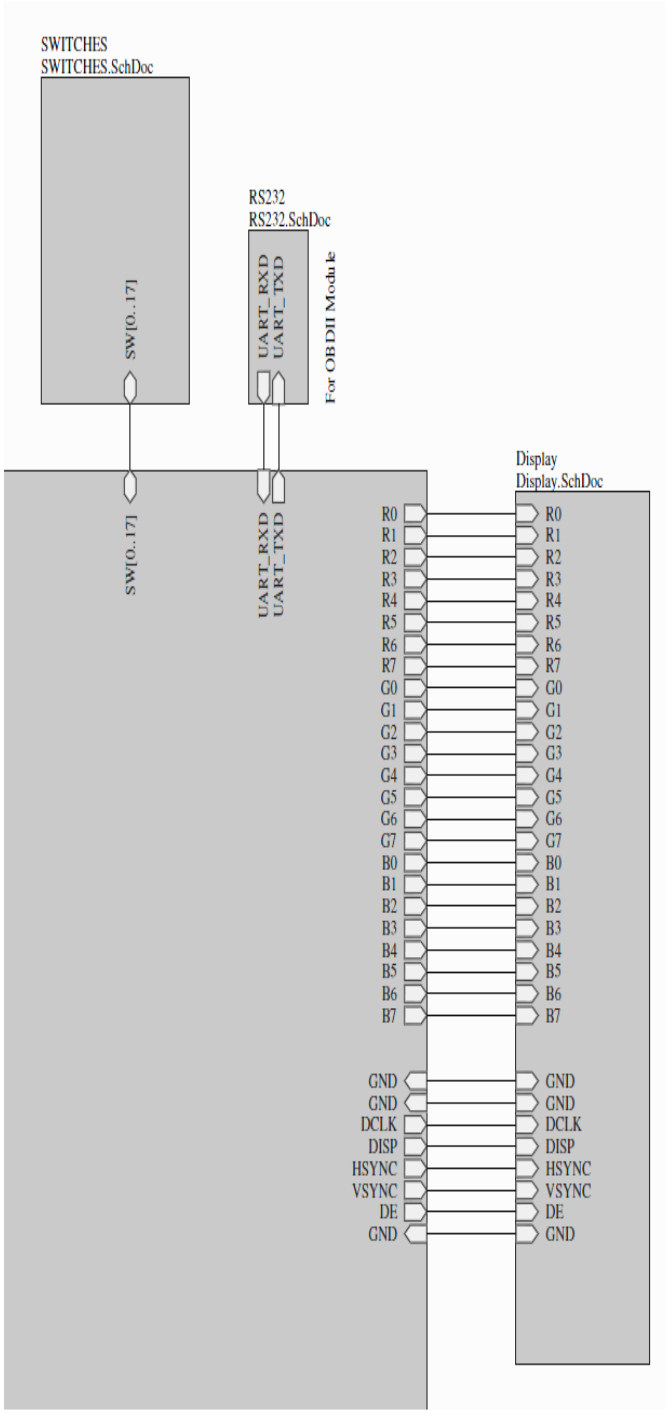

STORAGE_HEADER
STORAGE_HEADER.SchDoc
SPI_MISO/UART_BUSY
SPI_SCK
I2C_SDA
I2C_SCL
UART_RX/SPI_BUSY/I2C_BUSY
UART_TX
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
RESET
VBAT
+3.3V
ADC
ADC.SchDoc
FPGA
FPGA.SchDoc
D9
D8
D7
D6
LLC
D5
D4
D3
D2
D1
D0
IRQ
SCL
SDA
DEVADR
SPI_MISO/UART_BUSY
SPI_CLK
I2C_SDA
I2C_SCL
UART_RX/SPI_BUSY/I2C_BUSY
UART_TX
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
RESET
BOOTLOADER
BOOTLOADER_HEADER.SchDoc
MSEL0
MSEL1
DATA
DCLK
nCS
ASDI
nCEO
INIT_DONE
nSTATUS
nCONFIG
CONF_DONE
DCLK
nCE
FPGA_TDI
FPGA_TDO
FPGA_TCK
FPGA_TMS
DCLK
FPGA_TDO


ENCODE_DECODE
ENCODE_DECODE_SchDoc
HDATA_31
HDATA_30
HDATA_29
HDATA_28
HDATA_27
HDATA_26
HDATA_25
HDATA_24
HDATA_23
HDATA_22
HDATA_21
HDATA_20
HDATA_19
HDATA_18
HDATA_17
HDATA_16
HDATA_15
HDATA_14
HDATA_13
HDATA_12
HDATA_11
HDATA_10
HDATA_9
HDATA_8
HDATA_7
HDATA_6
HDATA_5
HDATA_4
HDATA_3
HDATA_2
HDATA_1
HDATA_0
VDATA_0
VDATA_1
VDATA_2
VDATA_3
VDATA_4
VDATA_5
VDATA_6
VDATA_7
VDATA_8
VDATA_9
VDATA_10
VDATA_11
FIELD/VSTROBE
VSYNC/VFRM
HSYNC/VRDY
VCLK
RCLK
IRQ
ACK
CS
TMS
TDI
TDO
TCK
DREQ0
DREQ1
DACK1
DACK0
ADDR_3
ADDR_2
ADDR_1
ADDR_0
MCLK
WE
RD
TRS
IRQJPEG


HDATA_31
HDATA_30
HDATA_29
HDATA_28
HDATA_27
HDATA_26
HDATA_25
HDATA_24
HDATA_23
HDATA_22
HDATA_21
HDATA_20
HDATA_19
HDATA_18
HDATA_17
HDATA_16
HDATA_15
HDATA_14
HDATA_13
HDATA_12
HDATA_11
HDATA_10
HDATA_9
HDATA_8
HDATA_7
HDATA_6
HDATA_5
HDATA_4
HDATA_3
HDATA_2
HDATA_1
HDATA_0
VDATA_0
VDATA_1
VDATA_2
VDATA_3
VDATA_4
VDATA_5
VDATA_6
VDATA_7
VDATA_8
VDATA_9
VDATA_10
VDATA_11
FIELD/VSTROBE
VSYNC/VFRM
HSYNC/VRDY
VCLK
RCLK
IRQJPEG
ACK
CS
TMS
TDI
TDO
TCK
DREQ0
DREQ1
DACK1
DACK0
ADDR_3
ADDR_2
ADDR_1
ADDR_0
MCLK
WE
RD
TRS
FPGA_TDO
DCLK
FPGA_TMS
FPGA_TCK
FPGA_TDO
FPGA_TDI
nCE
DCLK
CONF_DONE
nCONFIG
nSTATUS
INIT_DONE
nCEO
ASDI
nCS
DCLK
DATA
MSEL1
MSEL0
DEVADR
SDA
SCL
IRQ
D0
D1
D2
D3
D4
D5
LLC
D6
D7
D8
D9
FPGA
FPGA.SchDoc
SPI_MISO/UART_BUSY
SPI_CLK
I2C_SDA
I2C_SCL
UART_RX/SPI_BUSY/I2C_BUSY
UART_TX
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
RESET
+3.3V
SW[0..17]
UART_RXD
UART_TXD
GND
DE
VSYNC
HSYNC
DISP
DCLK
GND
GND
GND
B7
B6
B5
B4
B3
B2
B1
B0
G7
G6
G5
G4
G3
G2
G1
G0
R7
R6
R5
R4
R3
R2
R1
R0

POWER

POWER.SchDoc


SWITCHES
SWITCHES.SchDoc
SW[0..17]
RS232
RS232.SchDoc
UART_RXD
UART_TXD
For OBDII Module
Display
Display.SchDoc
SW[0..17]
UART_RXD
UART_TXD
R0
R1
R2
R3
R4
R5
R6
R7
G0
G1
G2
G3
G4
G5
G6
G7
B0
B1
B2
B3
B4
B5
B6
B7
GND
GND
DCLK
DISP
HSYNC
VSYNC
DE
GND

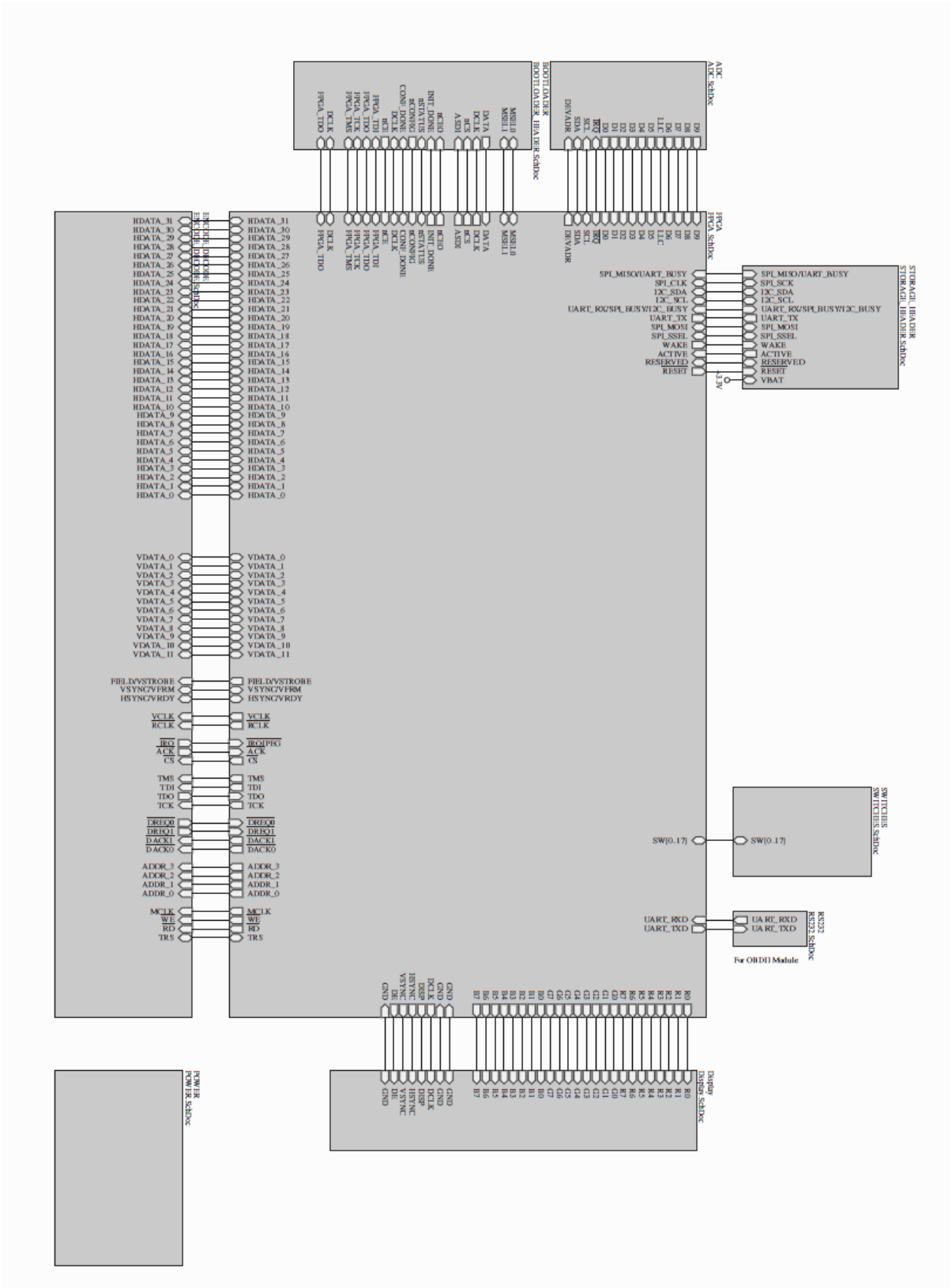
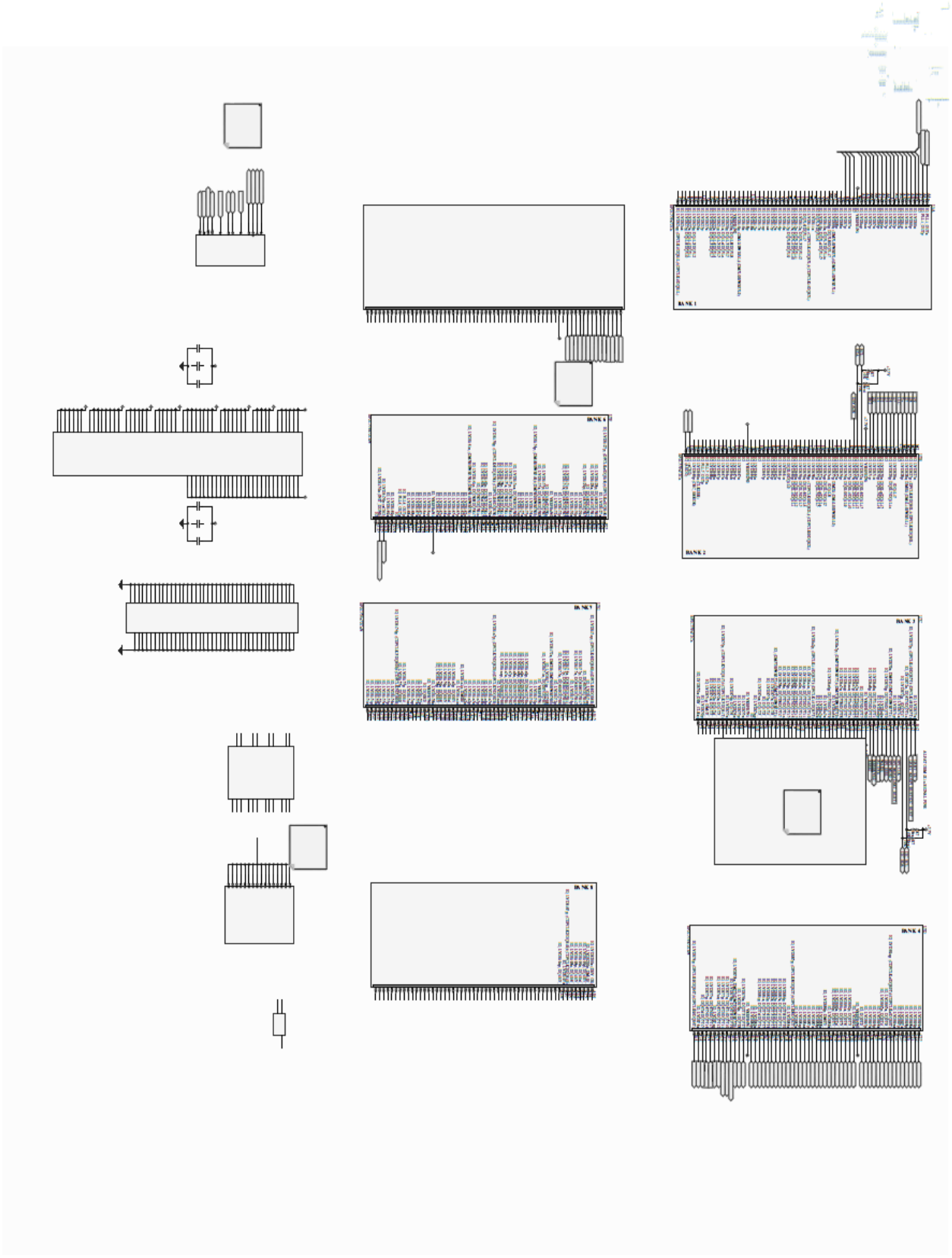
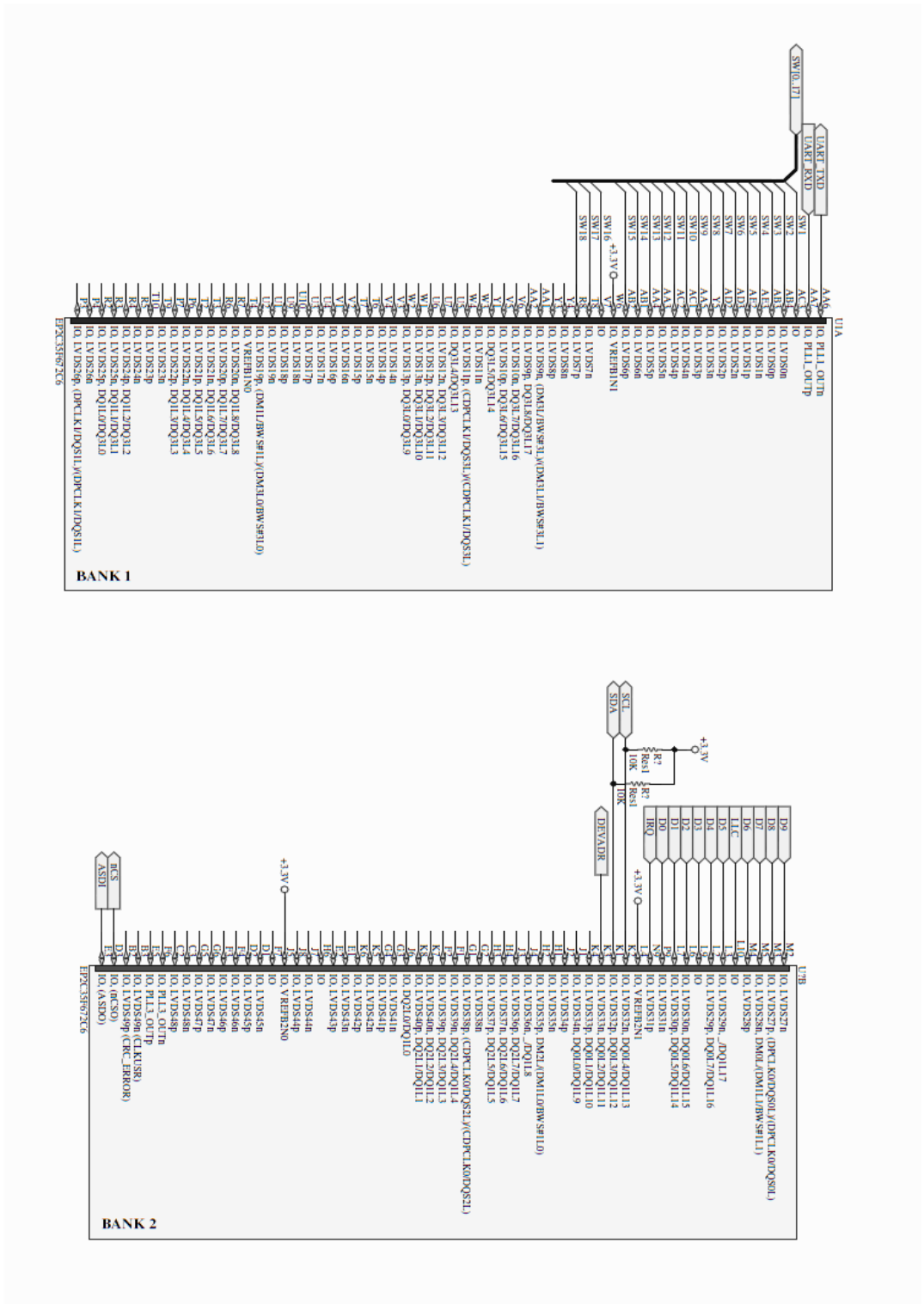
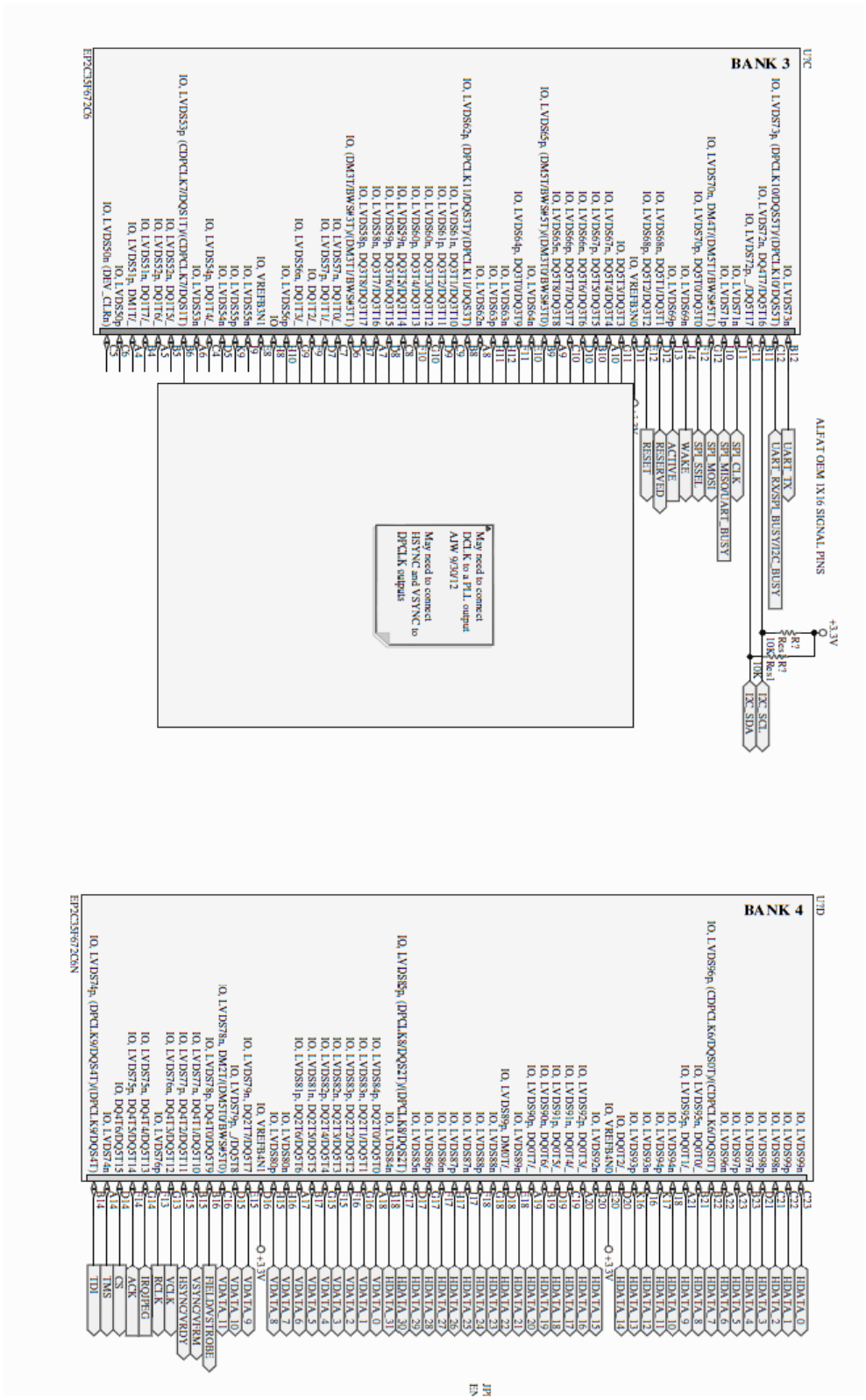
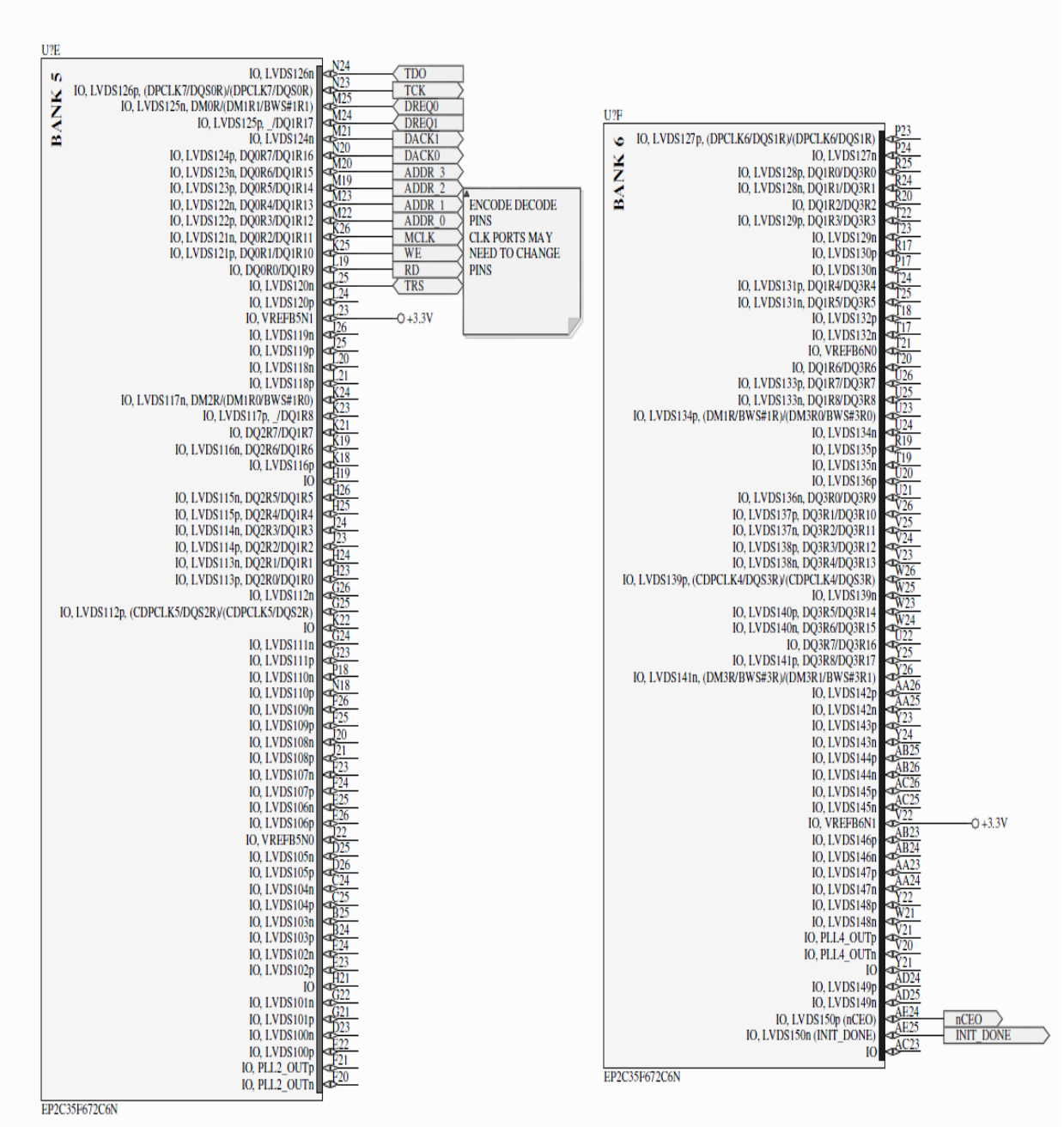

BOOTLOADER
BOOTLOADER_HEADER.SchDoc
ADC
ADC.SchDoc
FPGA
FPGA.SchDoc
STORAGE_HEADER
STORAGE_HEADER.SchDoc
SWITCHES
SWITCHES.SchDoc
RS232
RS232.SchDoc
For OBDII Module
Display
Display.SchDoc
POWER
POWER.SchDoc
SPI_MISO/UART_BUSY
SPI_CLK
I2C_SDA
I2C_SCL
UART_RX/SPI_BUSY/I2C_BUSY
UART_TX
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
RESET
VBAT
3.3V
SW[0..17]
UART_RXD
UART_TXD
HDATA_31
HDATA_30
HDATA_29
HDATA_28
HDATA_27
HDATA_26
HDATA_25
HDATA_24
HDATA_23
HDATA_22
HDATA_21
HDATA_20
HDATA_19
HDATA_18
HDATA_17
HDATA_16
HDATA_15
HDATA_14
HDATA_13
HDATA_12
HDATA_11
HDATA_10
HDATA_9
HDATA_8
HDATA_7
HDATA_6
HDATA_5
HDATA_4
HDATA_3
HDATA_2
HDATA_1
HDATA_0
VDATA_0
VDATA_1
VDATA_2
VDATA_3
VDATA_4
VDATA_5
VDATA_6
VDATA_7
VDATA_8
VDATA_9
VDATA_10
VDATA_11
FIELD/VSTROBE
VSYNC/VFRM
HSYNC/VRDY
VCLK
RCLK
IRQ
ACK
CS
TMS
TDI
TDO
TCK
DREQ0
DREQ1
DACK1
DACK0
ADDR_3
ADDR_2
ADDR_1
ADDR_0
MCLK
WE
RD
TRS
FPGA_TDO
DCLK
FPGA_TDI
FPGA_TMS
FPGA_TCK
nCE
CONF_DONE
nCONFIG
nSTATUS
INIT_DONE
nCEO
ASDI
nCS
DCLK
DATA
MSEL0
MSEL1
DATAADDR
SDA
SCL
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
GND
DE
HSYNC
VSYNC
DISP
DCLK
R0
R1
R2
R3
R4
R5
R6
R7
G0
G1
G2
G3
G4
G5
G6
G7
B0
B1
B2
B3
B4
B5
B6
B7

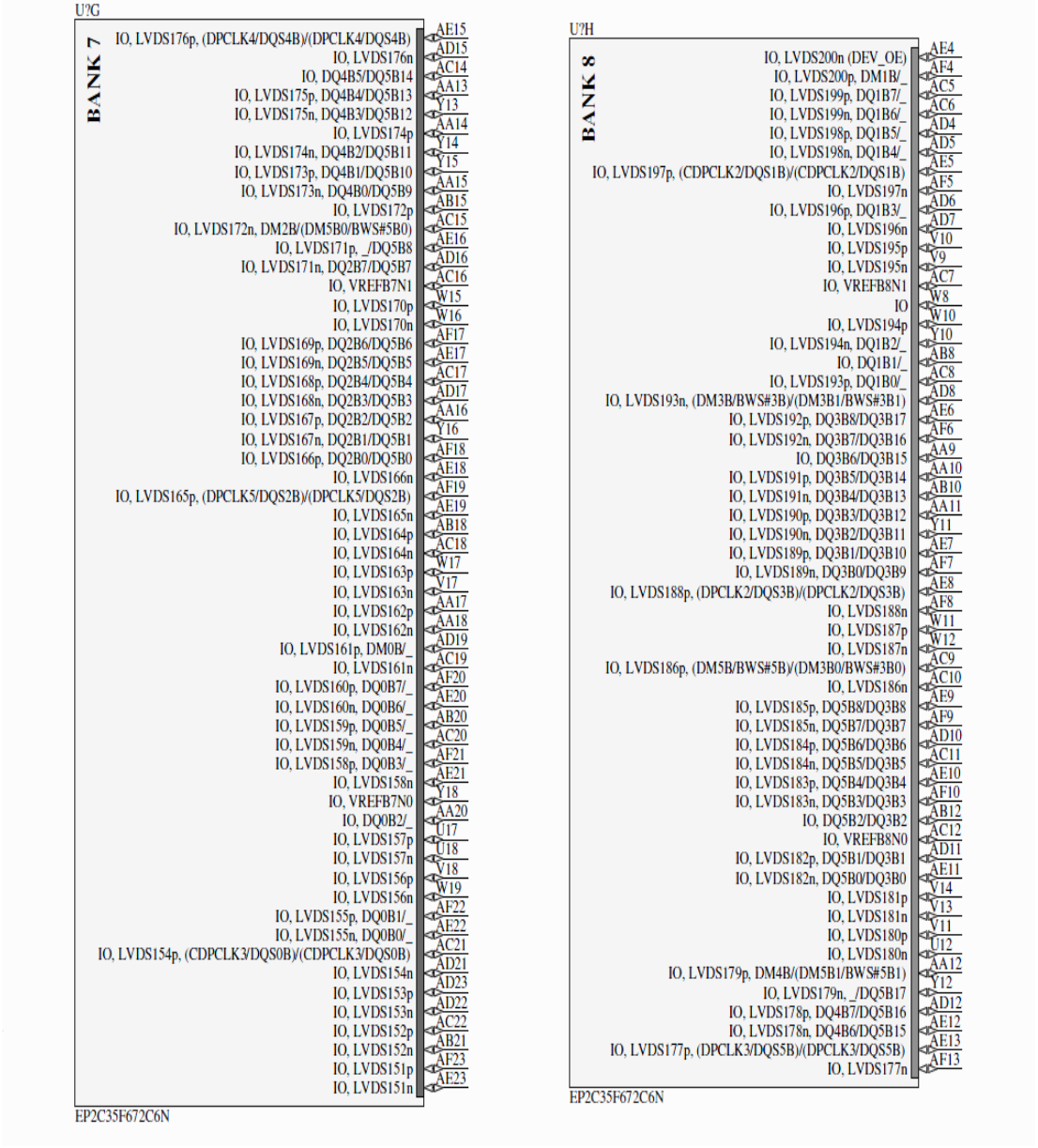
Figure 29: Top Level Schematic of Switches – RS232 – Display

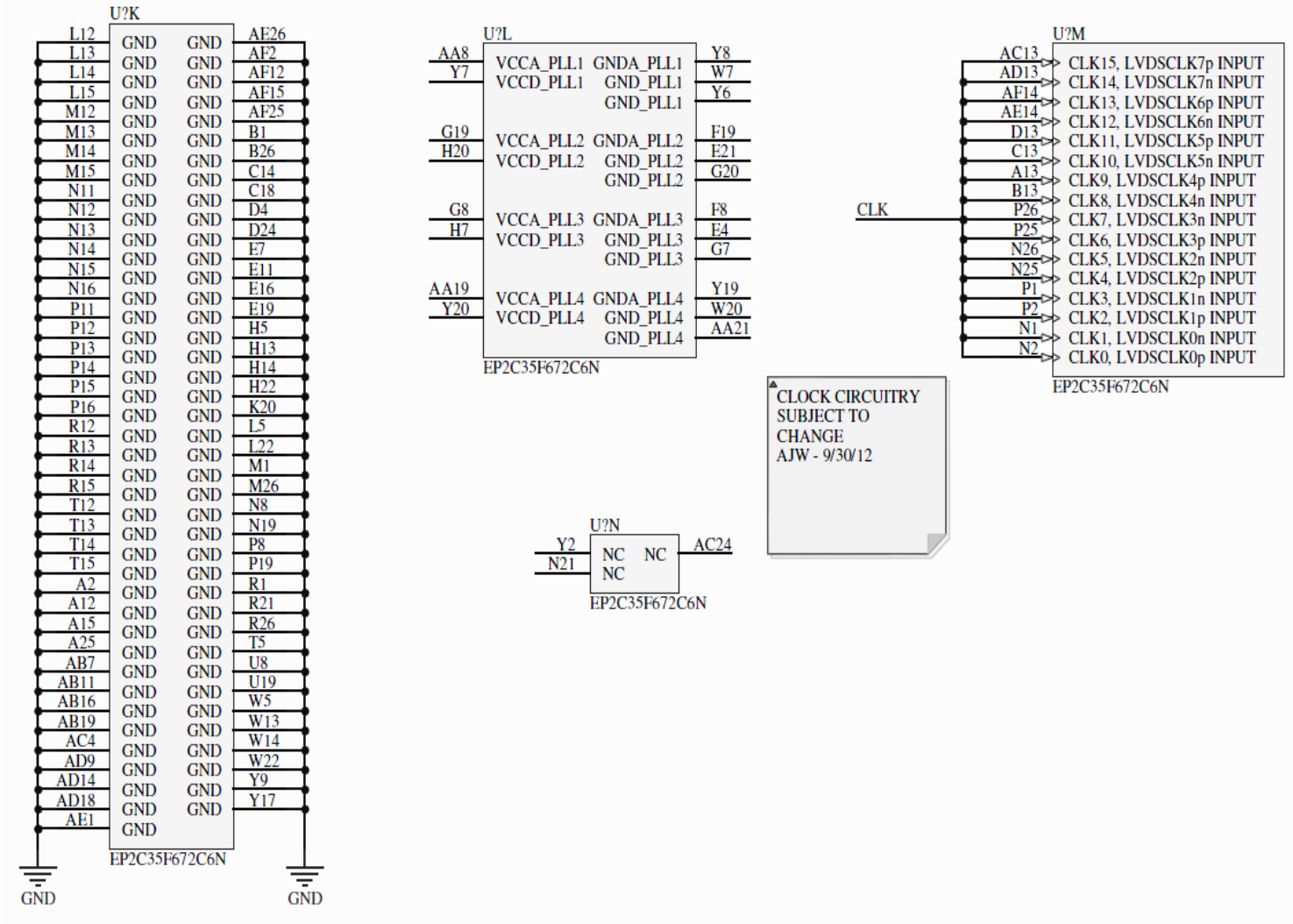


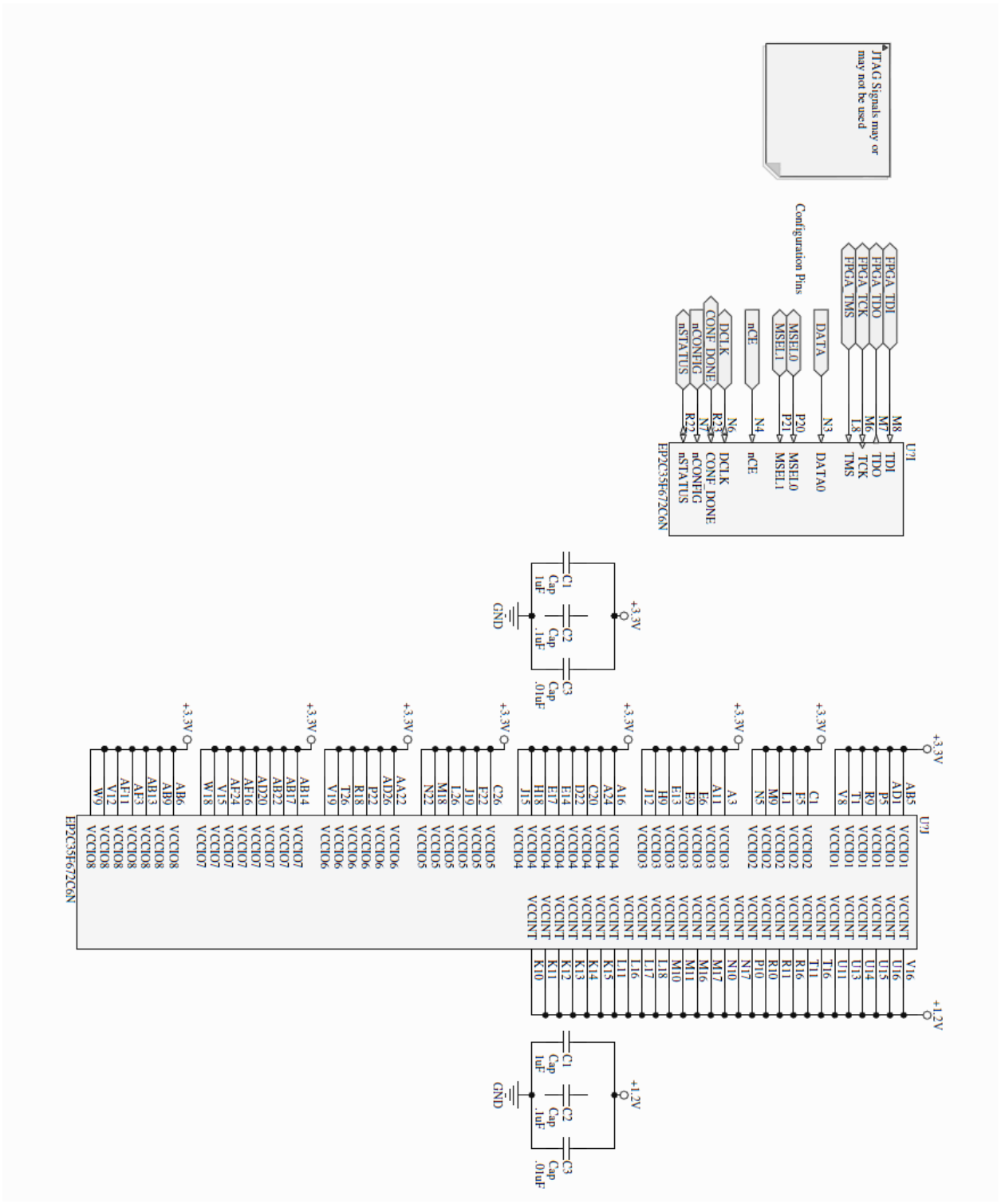

SW[0..17]
UART_TXD
UART_RXD
SW1
SW2
SW3
SW4
SW5
SW6
SW7
SW8
SW9
SW10
SW11
SW12
SW13
SW14
SW15
SW16
SW17
SW18
+3.3V
U1A
IO, PLL1_OUTn
IO, PLL1_OUTp
IO
IO, LVDS0n
IO, LVDS0p
IO, LVDS1n
IO, LVDS1p
IO, LVDS2n
IO, LVDS2p
IO, LVDS3n
IO, LVDS3p
IO, LVDS4n
IO, LVDS4p
IO, LVDS5n
IO, LVDS5p
IO, LVDS6n
IO, LVDS6p
IO, VREFB1N1
IO
IO, LVDS7n
IO, LVDS7p
IO, LVDS8n
IO, LVDS8p
IO, LVDS9n, (DM3L/BWS#3L)/(DM3L/BWS#3L1)
IO, LVDS9p, DQ3L8/DQ3L17
IO, LVDS10n, DQ3L7/DQ3L16
IO, LVDS10p, DQ3L6/DQ3L15
IO, LVDS11n
IO, LVDS11p, (CDPCLK1/DQS3L)/(CDPCLK1/DQS3L)
IO, DQ3L4/DQ3L13
IO, LVDS12n, DQ3L3/DQ3L12
IO, LVDS12p, DQ3L2/DQ3L11
IO, LVDS13n, DQ3L1/DQ3L10
IO, LVDS13p, DQ3L0/DQ3L9
IO, LVDS14n
IO, LVDS14p
IO, LVDS15n
IO, LVDS15p
IO, LVDS16n
IO, LVDS16p
IO, LVDS17n
IO, LVDS17p
IO, LVDS18n
IO, LVDS18p
IO, LVDS19n, (DM1L/BWS#1L)/(DM1L0/BWS#1L0)
IO, VREFB1N0
IO, LVDS20n, DQ1L8/DQ1L18
IO, LVDS20p, DQ1L7/DQ1L17
IO, LVDS21n, DQ1L6/DQ1L16
IO, LVDS21p, DQ1L5/DQ1L15
IO, LVDS22n, DQ1L4/DQ1L14
IO, LVDS22p, DQ1L3/DQ1L13
IO, LVDS23n
IO, LVDS23p
IO, LVDS24n, DQ1L2/DQ1L12
IO, LVDS24p, DQ1L1/DQ1L11
IO, LVDS25n, DQ1L0/DQ1L10
IO, LVDS25p, DQ1L9/DQ1L19
IO, LVDS26n, (CDPCLK0/DQS1L)/(CDPCLK0/DQS1L)
IO, LVDS26p, (DPCLK1/DQS1L)/(DPCLK1/DQS1L)
BANK 1
EP3C35F672C6
SCL
SDA
DEVADR
R?
10K
R?
10K
+3.3V
IRQ
D9
D8
D7
D6
LLC
D5
D4
D3
D2
D1
D0
U2B
IO, LVDS27n
IO, LVDS27p, (DPCLK0/DQS0L)/(DPCLK0/DQS0L)
IO, LVDS28n, DM0L/(DM1L)/BWS#1L1)
IO, LVDS28p
IO
IO, LVDS29n, /DQ1L17
IO, LVDS29p, DQ0L7/DQ1L16
IO
IO, LVDS30n, DQ0L6/DQ1L15
IO, LVDS30p, DQ0L5/DQ1L14
IO, LVDS31n
IO, LVDS31p
IO, VREFB2N1
IO, LVDS32n, DQ0L4/DQ1L13
IO, LVDS32p, DQ0L3/DQ1L12
IO, LVDS33n, DQ0L2/DQ1L11
IO, LVDS33p, DQ0L1/DQ1L10
IO, LVDS34n, DQ0L0/DQ1L9
IO, LVDS34p
IO, LVDS35n
IO, LVDS35p, DM2L/(DM1L0/BWS#1L0)
IO, LVDS36n, /DQ1L8
IO, LVDS36p, DQ2L7/DQ1L7
IO, LVDS37n, DQ2L6/DQ1L6
IO, LVDS37p, DQ2L5/DQ1L5
IO, LVDS38n
IO, LVDS38p, (CDPCLK0/DQS2L)/(CDPCLK0/DQS2L)
IO, LVDS39n, DQ2L4/DQ1L4
IO, LVDS39p, DQ2L3/DQ1L3
IO, LVDS40n, DQ2L2/DQ1L2
IO, LVDS40p, DQ2L1/DQ1L1
IO, LVDS41n
IO, LVDS41p
IO, LVDS42n
IO, LVDS42p
IO, LVDS43n
IO, LVDS43p
IO
IO, LVDS44n
IO, LVDS44p
IO, VREFB2N0
IO
IO, LVDS45n
IO, LVDS45p
IO, LVDS46n
IO, LVDS46p
IO, LVDS47n
IO, LVDS47p
IO, LVDS48n
IO, LVDS48p
IO, PLL3_OUTn
IO, PLL3_OUTp
IO, LVDS49n, (CLKUSR)
IO, LVDS49p, (CRC_ERROR)
IO, (#CSO)
IO, (ASDO)
+3.3V
#CS
ASDI
BANK 2
EP3C35F672C6


BANK 3
BANK 4
ALFAT OEM IX16 SIGNAL PINS
UART_TX
UART_RX/SPI_BUSY/I2C_BUSY
SPI_CLK
SPI_MISO/UART_BUSY
SPI_MOSI
SPI_SSEL
WAKE
ACTIVE
RESERVED
RESET
I2C_SCL
I2C_SDA
+3.3V
May need to connect DCLK to a PLL output
AJW 9/30/12
May need to connect HSYNC and VSYNC to DPCLK outputs
HDATA_0
HDATA_1
HDATA_2
HDATA_3
HDATA_4
HDATA_5
HDATA_6
HDATA_7
HDATA_8
HDATA_9
HDATA_10
HDATA_11
HDATA_12
HDATA_13
HDATA_14
HDATA_15
HDATA_16
HDATA_17
HDATA_18
HDATA_19
HDATA_20
HDATA_21
HDATA_22
HDATA_23
HDATA_24
HDATA_25
HDATA_26
HDATA_27
HDATA_28
HDATA_29
HDATA_30
HDATA_31
FIELD/VSTROBE
VSYNC/VFRM
HSYNC/VRDY
PCLK
IRQ/FIFG
ACK
CS
TMS
TDI
JPEN


U?E
BANK 5
IO, LVDS126n
IO, LVDS126p, (DPCLK7/DQS0R)/(DPCLK7/DQS0R)
IO, LVDS125n, DM0R/(DM1R1/BWS#1R1)
IO, LVDS125p, _/DQ1R17
IO, LVDS124n
IO, LVDS124p, DQ0R7/DQ1R16
IO, LVDS123n, DQ0R6/DQ1R15
IO, LVDS123p, DQ0R5/DQ1R14
IO, LVDS122n, DQ0R4/DQ1R13
IO, LVDS122p, DQ0R3/DQ1R12
IO, LVDS121n, DQ0R2/DQ1R11
IO, LVDS121p, DQ0R1/DQ1R10
IO, DQ0R0/DQ1R9
IO, LVDS120n
IO, LVDS120p
IO, VREFB5N1
IO, LVDS119n
IO, LVDS119p
IO, LVDS118n
IO, LVDS118p
IO, LVDS117n, DM2R/(DM1R0/BWS#1R0)
IO, LVDS117p, _/DQ1R8
IO, DQ2R7/DQ1R7
IO, LVDS116n, DQ2R6/DQ1R6
IO, LVDS116p
IO
IO, LVDS115n, DQ2R5/DQ1R5
IO, LVDS115p, DQ2R4/DQ1R4
IO, LVDS114n, DQ2R3/DQ1R3
IO, LVDS114p, DQ2R2/DQ1R2
IO, LVDS113n, DQ2R1/DQ1R1
IO, LVDS113p, DQ2R0/DQ1R0
IO, LVDS112n
IO, LVDS112p, (CDPCLK5/DQS2R)/(CDPCLK5/DQS2R)
IO
IO, LVDS111n
IO, LVDS111p
IO, LVDS110n
IO, LVDS110p
IO, LVDS109n
IO, LVDS109p
IO, LVDS108n
IO, LVDS108p
IO, LVDS107n
IO, LVDS107p
IO, LVDS106n
IO, LVDS106p
IO, VREFB5N0
IO, LVDS105n
IO, LVDS105p
IO, LVDS104n
IO, LVDS104p
IO, LVDS103n
IO, LVDS103p
IO, LVDS102n
IO, LVDS102p
IO
IO, LVDS101n
IO, LVDS101p
IO, LVDS100n
IO, LVDS100p
IO, PLL2_OUTp
IO, PLL2_OUTn
N24
N23
M25
M24
M21
N20
M20
M19
M23
M22
K26
K25
L19
L25
L24
L23
J26
J25
K24
K23
K21
K19
K18
H19
H26
H25
J24
J23
H24
H23
G26
G25
K22
G24
G23
P18
N18
F26
F25
E20
E21
F23
F24
F25
F26
E22
D25
D26
C24
C25
B25
B24
E24
E23
H21
G22
G21
D23
E22
F21
E20
TDO
TCK
DREQ0
DREQ1
DACK1
DACK0
ADDR_3
ADDR_2
ADDR_1
ADDR_0
MCLK
WE
RD
TRS
+3.3V
ENCODE DECODE PINS
CLK PORTS MAY NEED TO CHANGE PINS
EP2C35F672C6N
U?F
BANK 6
IO, LVDS127p, (DPCLK6/DQS1R)/(DPCLK6/DQS1R)
IO, LVDS127n
IO, LVDS128p, DQ1R0/DQ3R0
IO, LVDS128n, DQ1R1/DQ3R1
IO, DQ1R2/DQ3R2
IO, LVDS129p, DQ1R3/DQ3R3
IO, LVDS129n
IO, LVDS130p
IO, LVDS130n
IO, LVDS131p, DQ1R4/DQ3R4
IO, LVDS131n, DQ1R5/DQ3R5
IO, LVDS132p
IO, LVDS132n
IO, VREFB6N0
IO, DQ1R6/DQ3R6
IO, LVDS133p, DQ1R7/DQ3R7
IO, LVDS133n, DQ1R8/DQ3R8
IO, LVDS134p, (DM1R/BWS#1R)/(DM3R0/BWS#3R0)
IO, LVDS134n
IO, LVDS135p
IO, LVDS135n
IO, LVDS136p
IO, LVDS136n, DQ3R0/DQ3R9
IO, LVDS137p, DQ3R1/DQ3R10
IO, LVDS137n, DQ3R2/DQ3R11
IO, LVDS138p, DQ3R3/DQ3R12
IO, LVDS138n, DQ3R4/DQ3R13
IO, LVDS139p, (CDPCLK4/DQS3R)/(CDPCLK4/DQS3R)
IO, LVDS139n
IO, LVDS140p, DQ3R5/DQ3R14
IO, LVDS140n, DQ3R6/DQ3R15
IO, DQ3R7/DQ3R16
IO, LVDS141p, DQ3R8/DQ3R17
IO, LVDS141n, (DM3R/BWS#3R)/(DM3R1/BWS#3R1)
IO, LVDS142p
IO, LVDS142n
IO, LVDS143p
IO, LVDS143n
IO, LVDS144p
IO, LVDS144n
IO, LVDS145p
IO, LVDS145n
IO, VREFB6N1
IO, LVDS146p
IO, LVDS146n
IO, LVDS147p
IO, LVDS147n
IO, LVDS148p
IO, LVDS148n
IO, PLL4_OUTp
IO, PLL4_OUTn
IO
IO, LVDS149p
IO, LVDS149n
IO, LVDS150p (nCEO)
IO, LVDS150n (INIT_DONE)
IO
P23
P24
P25
R24
R20
T22
T23
R17
P17
T24
T25
R18
T17
T21
T20
U26
U25
U23
U24
R19
T19
U20
U21
V26
V25
V24
V23
W26
W25
W23
W24
T22
Y25
Y26
AA26
AA25
Y23
Y24
AB25
AB26
AC26
AC25
Y22
AB23
AB24
AA23
AA24
Y22
W21
V21
V20
Y21
AD24
AD25
AE24
AE25
AC23
+3.3V
nCEO
INIT_DONE
EP2C35F672C6N


U?G
BANK 7
IO, LVDS176p, (DPCLK4/DQS4B)/(DPCLK4/DQS4B) AE15
IO, LVDS176n AD15
IO, DQ4B5/DQ5B14 AC14
IO, LVDS175p, DQ4B4/DQ5B13 AA13
IO, LVDS175n, DQ4B3/DQ5B12 Y13
IO, LVDS174p AA14
IO, LVDS174n, DQ4B2/DQ5B11 Y14
IO, LVDS173p, DQ4B1/DQ5B10 Y15
IO, LVDS173n, DQ4B0/DQ5B9 AA15
IO, LVDS172p AB15
IO, LVDS172n, DM2B/(DM5B0/BWS#5B0) AC15
IO, LVDS171p, _/DQ5B8 AE16
IO, LVDS171n, DQ2B7/DQ5B7 AD16
IO, VREFB7N1 AC16
IO, LVDS170p W15
IO, LVDS170n W16
IO, LVDS169p, DQ2B6/DQ5B6 AF17
IO, LVDS169n, DQ2B5/DQ5B5 AE17
IO, LVDS168p, DQ2B4/DQ5B4 AC17
IO, LVDS168n, DQ2B3/DQ5B3 AD17
IO, LVDS167p, DQ2B2/DQ5B2 AA16
IO, LVDS167n, DQ2B1/DQ5B1 Y16
IO, LVDS166p, DQ2B0/DQ5B0 AF18
IO, LVDS166n AE18
IO, LVDS165p, (DPCLK5/DQS2B)/(DPCLK5/DQS2B) AF19
IO, LVDS165n AE19
IO, LVDS164p AB18
IO, LVDS164n AC18
IO, LVDS163p W17
IO, LVDS163n V17
IO, LVDS162p AA17
IO, LVDS162n AA18
IO, LVDS161p, DM0B/_ AD19
IO, LVDS161n AC19
IO, LVDS160p, DQ0B7/_ AF20
IO, LVDS160n, DQ0B6/_ AE20
IO, LVDS159p, DQ0B5/_ AB20
IO, LVDS159n, DQ0B4/_ AC20
IO, LVDS158p, DQ0B3/_ AF21
IO, LVDS158n AE21
IO, VREFB7N0 Y18
IO, DQ0B2/_ AA20
IO, LVDS157p U17
IO, LVDS157n U18
IO, LVDS156p V18
IO, LVDS156n W19
IO, LVDS155p, DQ0B1/_ AF22
IO, LVDS155n, DQ0B0/_ AE22
IO, LVDS154p, (CDPCLK3/DQS0B)/(CDPCLK3/DQS0B) AC21
IO, LVDS154n AD21
IO, LVDS153p AD23
IO, LVDS153n AD22
IO, LVDS152p AC22
IO, LVDS152n AB21
IO, LVDS151p AF23
IO, LVDS151n AE23
EP2C35F672C6N
U?H
BANK 8
IO, LVDS200n (DEV_OE) AE4
IO, LVDS200p, DM1B/_ AF4
IO, LVDS199p, DQ1B7/_ AC5
IO, LVDS199n, DQ1B6/_ AC6
IO, LVDS198p, DQ1B5/_ AD4
IO, LVDS198n, DQ1B4/_ AD5
IO, LVDS197p, (CDPCLK2/DQS1B)/(CDPCLK2/DQS1B) AE5
IO, LVDS197n AF5
IO, LVDS196p, DQ1B3/_ AD6
IO, LVDS196n AD7
IO, LVDS195p V10
IO, LVDS195n V9
IO, VREFB8N1 AC7
IO W8
IO, LVDS194p W10
IO, LVDS194n, DQ1B2/_ Y10
IO, DQ1B1/_ AB8
IO, LVDS193p, DQ1B0/_ AC8
IO, LVDS193n, (DM3B/BWS#3B)/(DM3B1/BWS#3B1) AD8
IO, LVDS192p, DQ3B8/DQ3B17 AE6
IO, LVDS192n, DQ3B7/DQ3B16 AF6
IO, DQ3B6/DQ3B15 AA9
IO, LVDS191p, DQ3B5/DQ3B14 AA10
IO, LVDS191n, DQ3B4/DQ3B13 AB10
IO, LVDS190p, DQ3B3/DQ3B12 AA11
IO, LVDS190n, DQ3B2/DQ3B11 Y11
IO, LVDS189p, DQ3B1/DQ3B10 AE7
IO, LVDS189n, DQ3B0/DQ3B9 AF7
IO, LVDS188p, (DPCLK2/DQS3B)/(DPCLK2/DQS3B) AE8
IO, LVDS188n AF8
IO, LVDS187p W11
IO, LVDS187n W12
IO, LVDS186p, (DM5B/BWS#5B)/(DM3B0/BWS#3B0) AC9
IO, LVDS186n AC10
IO, LVDS185p, DQ5B8/DQ3B8 AE9
IO, LVDS185n, DQ5B7/DQ3B7 AF9
IO, LVDS184p, DQ5B6/DQ3B6 AD10
IO, LVDS184n, DQ5B5/DQ3B5 AC11
IO, LVDS183p, DQ5B4/DQ3B4 AE10
IO, LVDS183n, DQ5B3/DQ3B3 AF10
IO, DQ5B2/DQ3B2 AB12
IO, VREFB8N0 AC12
IO, LVDS182p, DQ5B1/DQ3B1 AD11
IO, LVDS182n, DQ5B0/DQ3B0 AE11
IO, LVDS181p V14
IO, LVDS181n V13
IO, LVDS180p V11
IO, LVDS180n U12
IO, LVDS179p, DM4B/(DM5B1/BWS#5B1) AA12
IO, LVDS179n, _/DQ5B17 Y12
IO, LVDS178p, DQ4B7/DQ5B16 AD12
IO, LVDS178n, DQ4B6/DQ5B15 AE12
IO, LVDS177p, (DPCLK3/DQS5B)/(DPCLK3/DQS5B) AE13
IO, LVDS177n AF13
EP2C35F672C6N


U?K
L12 GND GND AE26
L13 GND GND AF2
L14 GND GND AF12
L15 GND GND AF15
M12 GND GND AF25
M13 GND GND B1
M14 GND GND B26
M15 GND GND C14
N11 GND GND C18
N12 GND GND D4
N13 GND GND D24
N14 GND GND E7
N15 GND GND E11
N16 GND GND E16
P11 GND GND E19
P12 GND GND H5
P13 GND GND H13
P14 GND GND H14
P15 GND GND H22
P16 GND GND K20
R12 GND GND L5
R13 GND GND L22
R14 GND GND M1
R15 GND GND M26
T12 GND GND N8
T13 GND GND N19
T14 GND GND P8
T15 GND GND P19
A2 GND GND R1
A12 GND GND R21
A15 GND GND R26
A25 GND GND T5
AB7 GND GND U8
AB11 GND GND U19
AB16 GND GND W5
AB19 GND GND W13
AC4 GND GND W14
AD9 GND GND W22
AD14 GND GND Y9
AD18 GND GND Y17
AE1 GND
EP2C35F672C6N
GND
GND
U?L
AA8 VCCA_PLL1 GNDA_PLL1 Y8
Y7 VCCD_PLL1 GND_PLL1 W7
GND_PLL1 Y6
G19 VCCA_PLL2 GNDA_PLL2 F19
H20 VCCD_PLL2 GND_PLL2 E21
GND_PLL2 G20
G8 VCCA_PLL3 GNDA_PLL3 F8
H7 VCCD_PLL3 GND_PLL3 E4
GND_PLL3 G7
AA19 VCCA_PLL4 GNDA_PLL4 Y19
Y20 VCCD_PLL4 GND_PLL4 W20
GND_PLL4 AA21
EP2C35F672C6N
U?M
CLK
AC13 CLK15, LVDSCLK7p INPUT
AD13 CLK14, LVDSCLK7n INPUT
AF14 CLK13, LVDSCLK6p INPUT
AE14 CLK12, LVDSCLK6n INPUT
D13 CLK11, LVDSCLK5p INPUT
C13 CLK10, LVDSCLK5n INPUT
A13 CLK9, LVDSCLK4p INPUT
B13 CLK8, LVDSCLK4n INPUT
P26 CLK7, LVDSCLK3n INPUT
P25 CLK6, LVDSCLK3p INPUT
N26 CLK5, LVDSCLK2n INPUT
N25 CLK4, LVDSCLK2p INPUT
P1 CLK3, LVDSCLK1n INPUT
P2 CLK2, LVDSCLK1p INPUT
N1 CLK1, LVDSCLK0n INPUT
N2 CLK0, LVDSCLK0p INPUT
EP2C35F672C6N
CLOCK CIRCUITRY
SUBJECT TO
CHANGE
AJW - 9/30/12
U?N
Y2 NC NC AC24
N21 NC
EP2C35F672C6N


JTAG Signals may or may not be used
Configuration Pins
FPGA_TDI
FPGA_TDO
FPGA_TCK
FPGA_TMS
DATA
MSEL0
MSEL1
nCE
DCLK
CONF_DONE
nCONFIG
nSTATUS
M8
M7
M6
L8
N3
P20
P21
N4
N6
R23
N7
R22
U?I
TDI
TDO
TCK
TMS
DATA0
MSEL0
MSEL1
nCE
DCLK
CONF_DONE
nCONFIG
nSTATUS
EP2C35F672C6N
+3.3V
GND
C1
Cap
1uF
C2
Cap
.1uF
C3
Cap
.01uF
U?J
AB5
AD1
P5
R9
T1
V8
C1
F5
L1
M9
N5
A3
A11
E6
E9
E13
H9
J12
A16
A24
C20
D22
E14
E17
H18
J15
C26
F22
J19
L26
M18
N22
AA22
AD26
P22
R18
T26
V19
AB14
AB17
AB22
AD20
AF16
AF24
V15
W18
AB6
AB9
AB13
AF3
AF11
V12
W9
V16
U16
U15
U14
U13
U11
T16
T11
R16
R11
R10
P10
N17
N10
M17
M16
M11
M10
L18
L17
L16
L11
K15
K14
K13
K12
K11
K10
VCCIO1
VCCIO2
VCCIO3
VCCIO4
VCCIO5
VCCIO6
VCCIO7
VCCIO8
VCCINT
+1.2V
EP2C35F672C6N

2.4 Simulations and Calculations

Color Conversion Calculations:

From Poynton's "Introduction to Digital Video" (p. 176, equations 9.6 and 9.7), the formula to convert RGB to YCbCr signal is,

$$\mathrm{YCbCr} = \mathrm{T} * \mathrm{RGB} + \mathrm{offset}$$

where,

$$\mathrm{T} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix}$$

and,

$$\mathrm{offset} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

This equation can be rewritten as,

$$\mathrm{RGB} = \mathrm{T}^{-1} * (\mathrm{YCbCr} - \mathrm{offset})$$

where,

$$\mathrm{T}^{-1} = \begin{bmatrix} 0.00456621 & 0 & 0.00625893 \\ 0.00456621 & -0.00153632 & -0.00318811 \\ 0.00456621 & 0.00791071 & 0 \end{bmatrix}$$

Color Conversion Simulation:

Using the derived equations, we implement our color conversion on MATLAB as follows,

```
function [ R, G, B , rgb] = YCbCr2RGB( Y, Cb, Cr )
%Function to convert from YCbCr to RGB used for simulation
    % Input checks
    if (Y>255 || Y<0 || Cb>255 || Cb<0 || Cr>255 || Cr<0)
        error('Invalid Input Range');
    end

    % Converting to fixed point
    max = 255;
    Y_r = (Y/max)*(2^7);
    Cb_r = (Cb/max)*(2^7);
    Cr_r = (Cr/max)*(2^7);

    % New fixed point inputs
    YCbCr = [Y_r; Cb_r; Cr_r];

    % T matrix and offset used for calculations
    T = [65.481 128.553 24.966; -37.797 -74.203 112; 112 -93.786
-18.214];
    offset = [16;128;128];
    % Inverting the T matrix
    Tinv = T^-1;

    % The final RGB map result
    rgb = Tinv*(YCbCr-offset);

    % Converting the fixed point to RGB results
    R = (rgb(1)/(2^7))*max;
    G = (rgb(2)/(2^7))*max;
    B = (rgb(3)/(2^7))*max;
```

Some results from this MATLAB code are as follows,

```
>> [R,G,B] = YCbCr2RGB(128, 128, 128)

R =

   -0.3560


G =

    1.0389


B =

   -0.5657
```

```
>> [R,G,B] = YCbCr2RGB(255, 128, 0)

R =

   -0.5772


G =

    2.0269


B =

    0.0142
```

```
>> [R,G,B] = YCbCr2RGB(255, 128, 64)

R =

   -0.1766


G =

    1.8229


B =

    0.0142
```

Data Rate:

Our data capture rate is limited by our storage rates. Maximum write speed for the storage module, ALFAT SoC is only 1400 Kbytes/s, so there will be a need for storage device speed-up and video capture down-sampling. The FUSB2805 transceiver can be soldered into the ALFAT SoC to increase the data rate transfer speed. On a FUSB2805, data can be transmitted and receive at high speed (480Mbps), full speed (12Mbps) and low speed (1.5Mbps) through a 12-bit (SDR) interface. In other words, 12Mbps is an achievable speed and data rate design calculations use this rate as a bench mark.

In this project, we use the MAX9526 as our ADC to convert Analog Video into digital signal. The MAX9526 is a 10-bit 4x oversampling (54Msps) ADC with true 10-but digital processing. To decrease bandwidth because of storage limitations, however, we will down-sample the video signals twelve times. We down-sample by 4 to achieve a video rate of 30 fps. After that, we further down-sample by 3 to achieve 10 fps. In other words, we will sample at a rate of,

$$\frac{54\ \text{Msps}}{12} = 4.5\ \text{Msps}$$

The FPGA gets the digital video signal from at this rate from the MAX9526 ADC, processes it, overlays it with the data obtained from the OBDII, and sends it to the ADV212 encoder. The ADV212 has the following data input rates:

Table 23. Maximum Pixel Data Input Rates (121-Ball Package)

Interface	Compression Mode	Input Format	Input Rate Limit Active Resolution (MSPS)[1]	Approx Min Output Rate, Compressed Data[2] (Mbps)	Approx Max Output Rate, Compressed Data[3] (Mbps)
HDATA	Irreversible	8-bit data	34	98	150
	Irreversible	10-bit data	34	98	150
	Irreversible	12-bit data	34	98	150
	Irreversible	16-bit data	34	98	150
	Reversible	8-bit data	30	98	150
	Reversible	10-bit data	24	98	150
	Reversible	12-bit data	20	98	150
	Reversible	14-bit data	17	98	150
VDATA	Irreversible	8-bit data	48	98	150
	Irreversible	10-bit data	48	98	150
	Irreversible	12-bit data	48	98	150
	Reversible	8-bit data	30	98	150
	Reversible	10-bit data	24	98	150
	Reversible	12-bit data	20	98	150

Figure 37: Maximum Pixel Data Input Rates for 121-Ball Package (ADV212 Datasheet, p. 32)

According to the table, for a VDATA irreversible 10-bit data, with an input of 48 Msps, it is guaranteed to have the output rate in the range of [98, 150] Mbps.

With our input coming in at a rate of 4.5Msps, the ratio of the actual input rate to the input rate limit is,

$$\frac{4.5}{48} = 0.09375$$

This gives us a new range of output rate of,

$$[98, 150]\text{Mbps} * 0.09375 = [9.2, 14]\ \text{Mbps}$$

OBDII Parameters:

The OBDII sends one or two bytes of information which then need additional calculations to determine the correct values of the parameters. These are done with the following equations, where A denotes the first byte of data and B denotes the second, if there is a second data byte.

Temperature:
Data = A – 40 (degrees C)
Fuel Pressure:
Data = A * 3 (kPa)
Engine RPM:
Data = ((A * 256) + B) / 4 (rpm)
Vehicle Speed:
Data = A (km/hr)
Throttle Position:
Data = (A * 100) / 255 (%)

You have five modules in your block diagram but only 3 modules in the requirements and verification section. There should be a 1-to-1 mapping between each module in your block diagram and a requirement/verification plan.

III. Requirements and Verification

3.1 Requirements

3.1.1 Requirements Summary:

1) **Input Module:** This module should successfully obtain the captured video from the camera and convert it into digital form using an ADC. Besides, this module also sends the OBD II data to the FPGA and control module without errors.
2) **Storage Module:** This module should successfully store the overlaid video without corrupting the file system at a reasonable frame rate.
3) **Power Module:** This module should supply enough steady power to the circuit.

3.1.2 Verifications Summary

1) **Input Module:** The video input can be tested simply by displaying the image from the camcorder onto a screen. The latest OBD II information will be stored in a register, so we can display the contents of the register onto the screen to be sure they are correct. If possible we will use an OBD II simulator to verify our measurements. The signal inputs and outputs will be verified with a logic analyzer to ensure accuracy.
2) **Display Module:** We will verify that the real-time video is displayed on our screen.
3) **Power Module:** We will do a simple check to see if the power supply supplies the expected voltage and that the circuit stays powered on during an extended period of time.

Your verifications lack detail. You need to explain exactly how you will test the requirement. How will you power the device during testing? What equipment will you use to probe the output? What is a "proper" output? What should the inputs to the module be? How will you supply these inputs?

3.1.3 Input Module

Requirements	Verifications
Camcorder: 1) Output video signal is analog SD NTSC-M signal MAX9526: 2) Output video is ITU-R.656 uncompressed digital video 3) Ensure that the ADC chip is powered at the correct wattage 4) Configure the MAX9526 correctly	Camcorder: 1) Ensure analog source, the camcorder, is powered on and output is enabled by: a) Power the camera and check that it outputs a proper signal MAX9526: 2) Check ADC chip power. 3) Check that the ADC is programmed

This needs more detail. How will you power the camera? How will you check its output? What is a proper output?

WIG-09555: 5) Input data signal is OBDII data from car 6) OBDII to ELM signaling is set to ISO 9141-2 for Asian make vehicles 7) Output data is OBDII info over RS-232 interface	properly. 4) Program the MAX9536 according to data sheet. WIG-09555: 5) Ensure car is powered on: 6) Check automotive accessory power outlet: a. Check that the outlet is outputting 12 V 7) Verify OBDII ISO 9141-2 signaling scheme is selected. 6) Make sure OBDII module is powered on and is set correctly. Store sent information in a register, and use the hex display on the FPGA to check that the correct data is received and stored

How will you check the ADC power? Using a multimeter? Oscilloscope? What inputs must the ADC be supplied with?

3.1.6 Display Module

Requirements	Verifications
ADV212: 1) Displays decompressed video equivalent to capture rate 2) Video output size maximum of 800X480 resolution 3) Ensure that the ADV212 is powered correctly 4) Configure the ADV212 correctly 5) ADV212 must be initiated properly K50DWN0-V1-F: 6) Ensure the correct power wattage is being sent to the display	ADV212: 1) Ensure interface clock is configured to proper clocking rate 2) Ensure JPEG2000 video is decompressed and scaled properly 3) Check that V_{DD} is between 1.425 and 1.575 V and that I_{DD} is between 55 and 65 mA using a multimeter. Also, we must check that V_{DDIO} is between 3.135 and 3.465 V and I_{DDIO} is between 40 and 50 mA. Finally, V_{IN} needs to be between -0.3 and 3.765 V and I_{DYN} must be between 280 and 320 mA. 4) This can be checked by communicating with the chip through a CPU via a free software codec such as Kakadu. 5) First, 0x400 must be written to the EIRQIE at address 0x5. Then, the IRQ pin will go low, and we will check that EIRQFLG[10] is set using the FPGA and a simple code. We will also use the FPGA to read the application ID to ensure that the chip was correctly initialized. K50DWN0-V1-F

What is the proper clocking rate for the display? How will you check the clocking rate? What equipment will you need?

	6) Use a multimeter to ensure that V_{CC} is between -0.3 and 5.0V and I_{CC} is between 25 and 35 mA. Also, V_{DD} must be between 3.0 and 3.6 V and I_{DD} has to be between 15 and 19 mA.

3.1.7 Power Module

Requirements	Verifications
Power: 1) Supply enough voltage to power on the board 2) Supply constant power to the board 3) The car voltage is stepped down correctly to 5V, 3.3V, 1.8V, 1.2V and 1.5V 4) All the individual components power on and work consistently	Power: 1) Car outlet is live 2) The board doesn't power off intermittently 3) A multimeter will be used to check the voltage output of each DC/DC converter to see if 5V, 3.3V, 1.8V, 1.2V and 1.5V are achieved 4) Ensure each component is connected to its correct voltage supply by checking the voltage before connecting it to a component.

The tolerance analysis is incomplete. You have described what tolerance range is acceptable for the ADC oscillator but you have not explained why the ADC oscillator is vital to your project. Moreover, you have not described how you will verify that this range of tolerances is acceptable.

3.2 Tolerance Analysis

Oscillator Frequencies: Verify that the ADC oscillator crystal has a frequency of 27 MHz, plus or minus 50 ppm.

3.3 Ethical Issues

The purpose of this project is to develop a data logger device for racer, which helps them to better manage the condition of the car while racing. With such function, our device helps increase the safety and health of the driver, which is consistent with the first code of the IEEE Code of Ethics:

1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;

One of the factors that cause car accidents is system defects. Our device helps prevent these kinds of accidents by giving drivers immediate information about the system s other car while also them with video of a driving car.

Throughout the development of the device, we will follow the third code closely, and only make claims and estimates based on real data acquired from our data logger.

3. to be honest and realistic in stating claims or estimates based on available data;

Working with a data logger device, the most importance factor is the accuracy of the information obtained from the system. We will be honest and will not falsify the data acquired from our test procedures.

After this project, we will have learned a great deal about various real-world industrial systems such as the On-board Diagnostic (OBD II) system, the SD Flash memory FAT 32 file system, or the ITU-R BT 656 protocol. This will improve our understanding of these technologies and their applications, and also improve our technical competence, as directed in the 5th and 6th codes of the IEEE Code of Ethics:

5. to improve the understanding of technology; its appropriate application, and potential consequences;

6. to maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations;

Furthermore, while working on the project, we will build an environment that promote engineer professionalism, which welcomes constructive and honest criticisms, acknowledges errors, assists peer workers with their professional and academic developments, and credits appropriate contributions, as cited in the 7th and 10th codes of the IEEE Code of Ethics:

7. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;

10. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

Following the 9th code of the IEEE Code of Ethics,

9. to avoid injuring others, their property, reputation, or employment by false or malicious action;

We will make sure that the data logger device developed from this project will offer the users with data that is as accurate as possible, and will not provide them with false information, in order to avoid damaging that could occur to the users' health, their car or other properties.

IV. Cost and Schedule

4.1 Cost Analysis

a. Labor

Name	Rate	Hours	Total	Total x 2.5
Tung Do	$50/hr	180	$9000	$22500
Nick Greenway	$50/hr	180	$9000	$22500
Andrew Wesly	$50/hr	180	$9000	$22500
Total				$67500

b. Parts

Description	#	Manufacturer	Vendor	Cost/unit	Total Cost
Encoder/Decoder	1	Analog Devices	Digikey	$48.984	$48.984
Development Board	1	Altera DE2	NA	$269	0 - Have
Interperter	1	Elm Electronics	ELM Electronics	$23.50	$23.50
Decoder	1	Maxim Integrated	Mouser	$7.97	$7.97
Converter	1	Sparkfun		$49.95	$49.95

You need to have part numbers in your bill of materials.

		Electronics			
Control Switches	6	C&K Compnents	Mouser	$6.25	$37.50
FPGA	1	Altera	Altera	$149.50	$149.50
Digilent Basys2 Development Board – used as bootloader	1	DigilentInc	DigilentInc	$100.00	0 - Have
Card writer	1	GHI Electronics	GHI Electronics	$59.95	$59.95
Display	1	Kentec	Newark	$65.22	$65.22

Total Labor Cost	**Total Parts Cost**	**Total Project Cost**
$67,500	$791.59	**$68,291.59**

A task can only have one person in charge of it. Therefore, you cannot have a task which is to be done by "All". Research is not sufficient for an entire week for one person. Neither is ordering parts. You should not have any "to be decided" weeks. You need to create a plan for the semester. If this plan changes, that is fine. But at the least you need to set out a course of action.

4.2 Schedule

Week	Task Description	Group Members
9/16	**Proposals Due**	
	Software design for storage and boot loader	Tung Do
	Software design for OBD II and ITU-R BT. 656	Nick Greenway
	Research Hardware	Andrew Wesly
9/23	**Design Reviews Sign-up Closes**	
	Finalize the schematic	All
9/30	**Design Reviews**	
	Order storage interface	Tung Do
	Test OBD II interface	Nick Greenway
	Board layout/order components	Andrew Wesly
10/7		
	Software development for Boot loader	Tung Do
	Software development for Video interface	Nick Greenway
	Submit board for fab	Andrew Wesly
10/14		
	Test boot loader	All
	Test video interface	Nick Greenway
	Assemble the boards	Andrew Wesly
10/21	**Individual Progress Reports Due**	
	Test overlaid video on storage device	Tung Do
	Verify correctness of OBD II data obtained	Nick Greenway
	Test assembled boards	Andrew Wesly
10/28		

	TBD	Tung Do
	TBD	Nick Greenway
	TBD	Andrew Wesly
11/4	**Mock-up Demos and Mock Presentation Sign-up closes**	
	Mock Presentation – Control Slides	Tung Do
	Mock Presentation – Test Slides	Nick Greenway
	Mock Presentation – Design Slides	Andrew Wesly
11/11	**Last Day to Request 1st Revision PCB Fabrication**	
	Final software testing	Tung Do
	Final software testing	Nick Greenway
	Assemble everything in an enclosure	Andrew Wesly
11/18	**Thanksgiving Break, Last day to Request Final Revision PCB fab**	
11/25	**Demo and Presentation Sign-up closes**	
	Final tests and verifications	Tung Do
	Final tests and verifications	Nick Greenway
	Final tests and verifications	Andrew Wesly
12/2	**Demos and Presentations**	
	Demo, Final Paper – Intro & Conclusion	Tung Do
	Presentation, Final Paper – Testing & Verification	Nick Greenway
	Demo, Final Paper – Design Procedure	Andrew Wesly
12/9	**Presentations, Checkout, Final Paper, Lab Notebooks**	
	Final Paper & Notebook review and hand-in	Tung Do
	Final Paper & Notebook review and hand-in	Nick Greenway
	Final Paper & Notebook review and hand-in	Andrew Wesly

4.1 Contingency Plan

- In case of exorbitant pcb costs or a crunch for time the following contingency plan may be executed to deliver on proposed Automotive Racing Video Data Logger.

- Replace FPGA IC, MAX9526 ADC IC, RS232 MODULE, and CONTROL switches with Altera DE2 board.

- Modify power supply module to supply 9V to development board

- Connect ALFAT module to board, OBDII module to board

- Create TFT display connections

- Possibly implement JPEG2000 encoder/decoder.

- Frame rates may be as low as 1fps

References:

78SR 2 Amp Series, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/meters/mpm_78sr-2a_a00.pdf

ADV212, Rev. B, Analog Devices, Norwood, MA, 2010. [Online]. Available: http://www.analog.com/static/imported-files/data_sheets/ADV212.pdf

ALFAT SoC Processor User Manual, Rev. 1.12, GHI Electronics, Macomb Township, MI, 2012. [Online]. Available: http://www.ghielectronics.com/downloads/ALFAT/ALFAT%20SoC%20Processor%20User%20Manual.pdf

BT.656 Video Interface for ICs, Intersil, Milpitas, CA, 2002. [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/an97/an9728.pdf

Charles A. Poynton, *A Technical Introduction to Digital Video*, John Wiley & Sons, Inc., 1996, p. 175-176

Cyclone II Device Handbook, Volume 1, Altera, San Jose, CA, 2008. [Online]. Available: http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf

Cyclone II EP2C35 PCI Development Board Reference Manual, Version 1, Altera, San Jose, CA, 2005. [Online]. Available: http://www.altera.com/literature/manual/rm_pci_dev_bd_cyclone2_ed.pdf

Digilent Basys 2Board Reference Manual, Digilent, Pullman, WA, 2010. [Online]. Available: https://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf

ELM327, ELM, Toronto, [Online]. Available: http://elmelectronics.com/DSheets/ELM327DS.pdf

FS-K50DWN0-V1-F, Rev. 2, Kentec Display, 2011. [Online]. Available: http://www.kentecdisplay.com/uploads/soft/Products_spec/K50DWN0-V1-F-01_KED_.pdf

IEEE Code of Ethics [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

MAX9526, Rev. 3, Maxim Integrated, San Jose, CA, [Online]. Available: http://datasheets.maximintegrated.com/en/ds/MAX9526.pdf

MPDTY Series, Murata, Mansfield, MA, [Online]. Available: http://search.murata.co.jp/Ceramy/image/img/A14X/M07E1.pdf

NDTS Series, Murata, Mansfield, MA, 2012 [Online]. Available: http://www.murata-ps.com/data/power/ncl/kdc_ndts.pdf

OBD-II UART [Online]. Available: https://www.sparkfun.com/products/9555

Recommendation ITU-R BT.601-5: Studio Encoding Parameters of Digital Television for Standard 4:3 and Widescreen 16:9 Aspect Ratios, 1995. [Online]. Available: http://www.intersil.com/content/dam/Intersil/documents/an97/an9728.pdf

Single Output LSN-10A Models, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/power/lsn10a-d5.pdf

Single Output LSN-W3 Models, Murata, Mansfield, MA, 2009. [Online]. Available: http://www.murata-ps.com/data/power/lsn16a-w3.pdf